

Multidimensional Signal, Image, and Video Processing and Coding

Multidimensional Signal, Image, and Video Processing and Coding

Second Edition

John W. Woods

*Rensselaer Polytechnic Institute
Troy, New York*



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Academic Press is an imprint of Elsevier



Academic Press is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA
The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, UK

© 2012 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

MATLAB[®] is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB[®] software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB[®] software.

Library of Congress Cataloging-in-Publication Data

Woods, John W. (John William), 1943-

Multidimensional signal, image, and video processing and coding / John W. Woods.—2nd ed.
p. cm.

Includes bibliographical references.

ISBN 978-0-12-381420-3

1. Image processing—Digital techniques. 2. Signal processing—Digital techniques. 3. Video compression—Standards. 4. Digital video. I. Title.

TA1637.W67 2012

621.382'2—dc22

2011002927

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

For information on all Academic Press publications
visit our website at www.elsevierdirect.com

Typeset by: diacriTech, India

Printed in the United States of America

11 12 13 14 9 8 7 6 5 4 3 2 1

Working together to grow
libraries in developing countries

www.elsevier.com | www.bookaid.org | www.sabre.org

ELSEVIER

BOOK AID
International

Sabre Foundation

Preface

This is a textbook for a first- or second-year graduate course for electrical and computer engineering (ECE) students in the area of digital image and video processing and coding. The course might be called Digital Image and Video Processing (DIVP) or some such, and has its heritage in the signal processing and communications areas of ECE. The relevant image (and video) processing problems can be categorized as image-in/image-out, rather than image-in/analysis-out types of problems. Though for this second edition, we have broadened our coverage by adding introductory material on image analysis.

The required background is an undergraduate digital signal processing (DSP) course and a course in probability. In addition, some knowledge of random processes and information theory is required for some sections. So in this second edition, we have added introductory material on random processes and information theory in chapter appendices. In addition to its role as a graduate text, the book is also suitable for self-study by graduate engineers with a communications and signal processing background.

The DIVP course at Rensselaer Polytechnic Institute has been offered for the last 20 years, now on an alternate year basis, having started as a course in multidimensional DSP. Over the years we brought in an emphasis first on image processing and coding, and then on video, and finally, some coverage of network transmission of video. The book, as well as the DIVP course, starts out with two-dimensional (2-D) signal processing theory, including 2-D systems, partial difference equations, Fourier and Z-transforms, filter stability, discrete transforms such as DFT and DCT and their fast algorithms, ending up with 2-D or spatial filter design. We also introduce the sub-band/wavelet transform (SWT) here, along with coverage of the DFT and DCT. This material is contained in the first five chapters and constitutes the first part of the book. However, there is also a later chapter on 3-D and spatiotemporal signal processing, strategically positioned between the image and the video processing chapters.

The second part of the book, comprising the remaining seven chapters, covers image and video processing and coding, along with the introduction to video transmission on networks. We start out with image perception and sensing, followed by basic filters for image processing, and then treat image enhancement and image analysis. This is followed by four individual chapters on estimation/restoration and source coding, first for images and then for video. The final chapter on network transmission summarizes basic networking concepts and then goes on to consider effects of packet loss and their amelioration.

This paragraph and the next provide detailed chapter information. Starting out the first part, Chapter 1 introduces 2-D systems and signals along with the BIBO stability concept, Fourier transform, and spatial convolution. Chapter 2 covers sampling and considers both rectangular and general regular sampling patterns, e.g., diamond and hexagonal sample patterns. Chapter 3 introduces 2-D difference equations and the Z transform including recursive filter stability theorems. Chapter 4 treats the discrete

Fourier and cosine transforms along with their fast algorithms and briefly covers 2-D sectioned-convolution. We also introduce the ideal subband/wavelet transform (SWT) here, postponing its design problem to the next chapter. Chapter 5 covers 2-D filter design, mainly through the separable and circular window method, but also introducing the problem of 2-D recursive filter design, along with some coverage of general or fully recursive filters.

The second part of the book, the part on image and video processing and coding, starts out with Chapter 6, which presents basic concepts in human visual perception and color space, as well as basic coverage of common image sensors and displays. Chapter 7 first covers the basic image processing filters: box, Prewitt, and Sobel, and then employs them for simple image enhancement. The chapter also includes basic image analysis, including edge detection and linking, image segmentation, object detection, and template matching.

Chapter 8 covers image estimation and restoration, including adaptive or inhomogeneous approaches, and an introduction to image- and blur-model parameter identification via the EM algorithm. We also include material on compound Gauss-Markov models and their MAP estimation via simulated annealing. We look at new approaches such as Gaussian scale mixtures and nonlocal estimators. Chapter 9 covers image compression built up from the basic concepts of transform, scalar and vector quantization, and variable-length coding. We cover basic DCT coders and also include material on fully embedded coders such as EZW, SPIHT, and EZBC and introduce the main concepts of the JPEG 2000 standard. Then Chapter 10 on three-dimensional and spatiotemporal or multidimensional signal processing (MDSP) extends the 2-D concepts of Chapters 1–5 to the 3-D or spatiotemporal case of video. Also included here are rational system models and spatiotemporal Markov models culminating in a spatiotemporal reduced-update Kalman filter. Next, Chapter 11 studies motion estimation including block matching, variable size block matching, optical flow, and mesh-based matching, and the techniques of motion compensation. Applications include motion-compensated Kalman filtering, frame-rate change, and deinterlacing. The chapter ends with the Bayesian approach to joint motion estimation and segmentation, and a new section on super-resolution. Chapter 12 covers video compression with both hybrid and spatiotemporal transform approaches and includes coverage of video coding standards such as MPEG 2 and H.264/AVC. This second edition includes coverage on the new scalable video standard H.264/SVC and the 3-D stereoscopic standard H.264/MVC. We include coverage of highly scalable coders based on the motion-compensated temporal filter (MCTF).

Finally, Chapter 13 is devoted to video on networks, starting out with network fundamentals to make the chapter accessible to signal processing and communication graduate students without a networking background. The chapter then proceeds to robust methods for network video. We discuss error-concealment features in H.264/AVC. We include methods of error concealment and robust scalable approaches using MCTF and embedded source coding. We present multiple description coding in combination with forward error correction as an efficient way to deal

with packet losses. This second edition now includes basic introductory material for practical network coding. We look at combinations of multiple description coding and network coding for efficient heterogeneous multicast.

This book also has an associated Web site (<http://elsevierdirect.com/9780123814203>) that contains many short MATLAB programs that complement examples and exercises on MDSP. There is also a .pdf document on the site that contains high-quality versions of all the figures and images in the book. There are numerous short video clips showing applications in video processing and coding. The Web site has a copy of the *vidview* video player for playing .yuv video files on a Windows PC. Other video files can generally be decoded and played by the commonly available media decoder/players. Also included is an illustration of effect of packet loss on H.264/AVC coded bitstreams. (Your media decoder/player would need an H.264/AVC decoder component to play these, however, some .yuv files are included here in case it doesn't.)

This textbook can be utilized in several ways depending on the course level and desired learning objectives. One path is to first cover Chapters 1–5 on MDSP, and then go on to Chapters 6 through 9 to cover image processing and coding, followed by some material on video processing and coding from later chapters, and this is how we have most often used it at Rensselaer. Alternatively, after Chapters 1–5, one could go on to image and video processing in Chapters 6–8 and 10–11. Or, and again after covering Chapters 1–5, go on to image and video compression in Chapter 6–7, 9–10, and 12. The network transmission material from Chapter 13 could also be included, time permitting. To cover the image and video processing and coding in Chapters 6–12 in a single semester, some significant sampling of the first five chapters would probably be needed. One approach may be to skip (or very lightly cover) Chapter 3 on 2-D systems and Z transforms and Chapter 5 on 2-D filter design, but cover Chapters 1, 2, and part of Chapter 4. Still another possibility is to cover Chapters 1 and 2, and then move on to Chapters 6–12, introducing topics from Chapters 3–5 only as needed. An online solutions manual will be made available to instructors at <http://textbooks.elsevier.com> with completion of registration in the Electronics and Electrical Engineering subject area.

John W. Woods

*Rensselaer Polytechnic Institute
Troy, New York
Spring 2011*

ACKNOWLEDGMENTS

I started out writing the first edition of this book while teaching from the textbook *Two-Dimensional Digital Image Processing* by Jae Lim, and readers familiar with that book will certainly see a similarity to mine, especially in the first five

chapters. Thanks to colleagues Janusz Konrad, Aggelos Katsaggelos, Edward Delp, Siwei Lyu, and Dirk-Jan Kroon for providing examples. Thanks also to Fure-Ching Jeng, Shih-Ta Hsiang, Seung-Jong Choi, T. Naveen, Peisong Chen, Anil Murching, Allesandro Dutra, Ju-Hong Lee, Yongjun Wu, Eren Soyak, Yufeng Shan, Adarsh Ramasubramonian, and Han Huang. Special thanks to former student Ivan Bajić for contributing Section 13.2. Thanks also to my wife Harriet, who has cheerfully put up with this second edition.

Two-Dimensional Signals and Systems

1

This chapter sets forth the main concepts of two-dimensional (2-D) signals and systems as extensions of the linear systems concepts of 1-D signals and systems. We concentrate on the discrete-space case of digital data, including the corresponding 2-D Fourier transform. We also introduce the continuous-space Fourier transform to deal with angular rotations and prove the celebrated projection-slice theorem of computer tomography. In later chapters we will encounter the central role of motion compensation in image sequences and video, where motion is often not an exact number of pixels, which makes the interplay of discrete parameters and continuous parameters quite important. While this first chapter focuses on discrete space, Chapter 2 focuses on sampling of 2-D continuous-space functions.

1.1 TWO-DIMENSIONAL SIGNALS

A scalar 2-D signal $s(n_1, n_2)$ is mathematically a complex *bi-sequence*, or a mapping of the 2-D integers into the complex plane. Our convention is that the signal s is defined for all finite values of its integer arguments n_1, n_2 using zero padding as necessary. Occasionally we will deal with finite extent signals, but we will clearly say so in such instances. We will adopt the simplified term *sequence* over the more correct bi-sequence.

A simple example of a 2-D signal is the *impulse* $\delta(n_1, n_2)$, defined as follows:

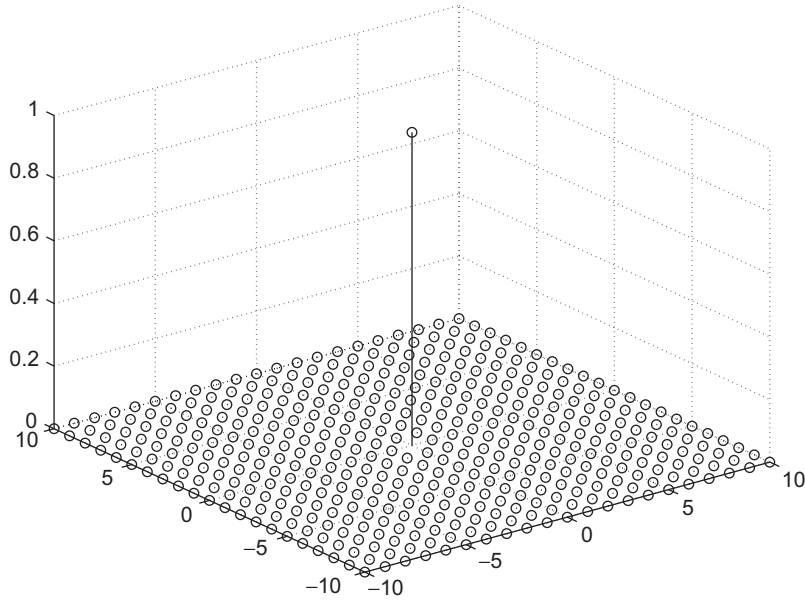
$$\delta(n_1, n_2) \triangleq \begin{cases} 1 & \text{for } (n_1, n_2) = (0, 0), \\ 0 & \text{for } (n_1, n_2) \neq (0, 0), \end{cases}$$

a portion of which is plotted in [Figure 1.1–1](#).

A general 2-D signal can be written as an infinite sum over shifted impulses, which will be found useful later:

$$x(n_1, n_2) = \sum_{k_1, k_2} x(k_1, k_2) \delta(n_1 - k_1, n_2 - k_2), \quad (1.1-1)$$

where the summation is taken over all integer pairs (k_1, k_2) . By the definition of the 2-D impulse, for each point (n_1, n_2) , there is exactly one term on the right side that is

**FIGURE 1.1–1**

MATLAB plot of 2-D spatial impulse bi-sequence $\delta(n_1, n_2)$.

nonzero, the term $x(n_1, n_2) \cdot 1$, and hence the result (1.1–1) is correct. This equality is called the *shifting representation* of the signal x .

Another basic 2-D signal is the impulse line, e.g., a straight line at 45° ,

$$\delta(n_1 - n_2) = \begin{cases} 1, & n_1 = n_2, \\ 0, & \text{else.} \end{cases}$$

A portion of this line is sketched in Figure 1.1–2.

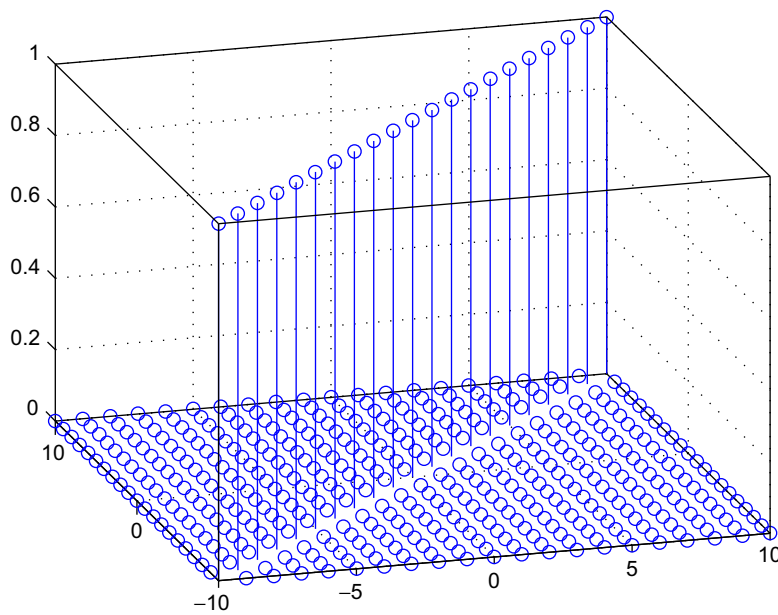
We can also consider line impulses at 0° , 90° , and -45° in discrete space, but other angles give “gaps in the line.” Toward the end of this chapter, we will look at continuous-space signals; there the line impulse can be at any angle.

Another basic 2-D signal class is step functions, perhaps the most common of which is the first-quadrant *unit step function* $u(n_1, n_2)$, defined as follows:

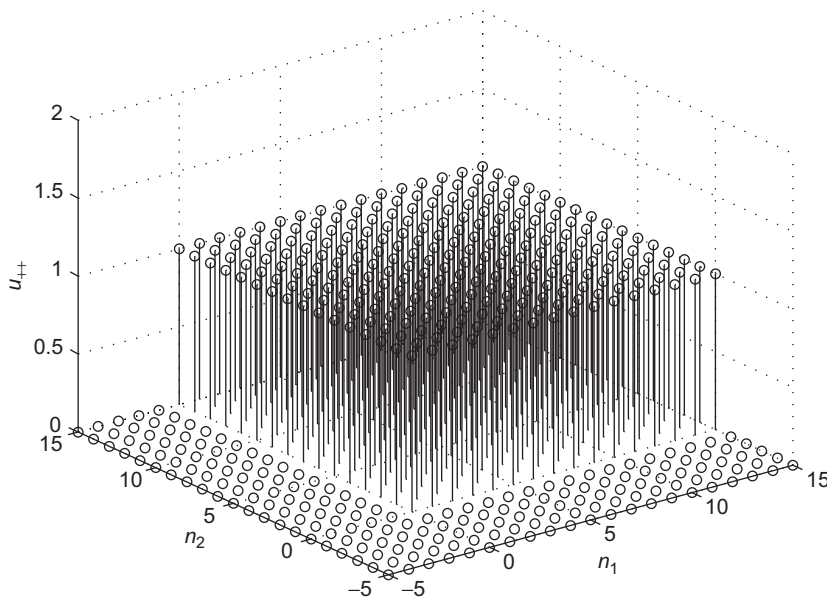
$$u(n_1, n_2) \triangleq \begin{cases} 1, & n_1 \geq 0, n_2 \geq 0, \\ 0, & \text{elsewhere.} \end{cases}$$

A portion of this bi-sequence is plotted in Figure 1.1–3.

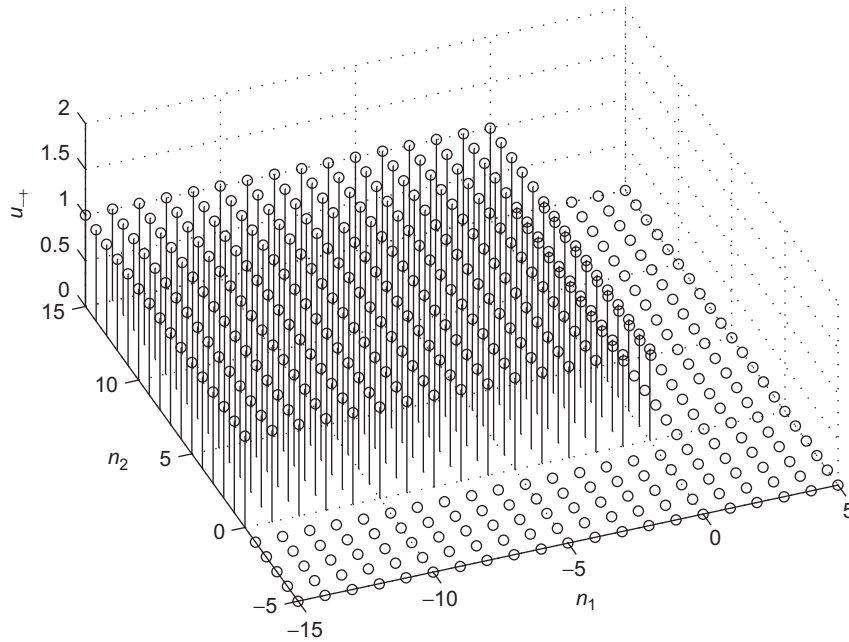
Actually, in two dimensions, there are several step functions of interest. The one shown in Figure 1.1–3 is called the first quadrant unit step function and is more generally denoted as $u_{++}(n_1, n_2)$.

**FIGURE 1.1-2**

A portion of the unit impulse line $\delta(n_1 - n_2)$ at 45° .

**FIGURE 1.1-3**

A portion of the unit step bi-sequence $u(n_1, n_2)$.

**FIGURE 1.1–4**

A portion of the second quadrant unit step bi-sequence $u_{-+}(n_1, n_2)$.

We will find it convenient to use the word *support* to denote the set of all argument values for which the function is nonzero. In the case of the first quadrant unit step $u_{++}(n_1, n_2)$, this becomes

$$\text{supp}(u_{++}) = \{n_1 \geq 0, n_2 \geq 0\}.$$

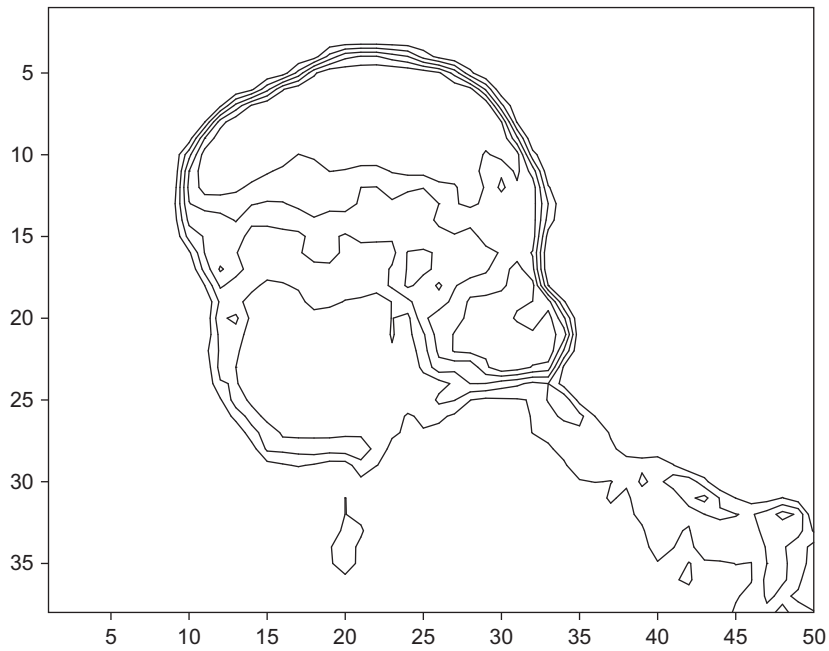
Three other unit step functions can be defined for the other three quadrants. They are denoted u_{-+} , u_{+-} , and u_{--} , with support on the second, fourth, and third quadrants, respectively. A plot of a portion of the second quadrant unit step is shown in Figure 1.1–4. We often write simply $u_{++} = u$ for the first quadrant unit step. Later, we will also find it convenient to talk about halfplane support unit steps defined analogously.

A real example of a finite support 2-D sequence is the image *Eric* shown in three different ways in Figures 1.1–5 through 1.1–7. The first, Figure 1.1–5, is a contour plot of *Eric*, a 100×76 -pixel, 8-bit grayscale image. The second, Figure 1.1–6, is a perspective (or mesh) plot. The third, Figure 1.1–7, is an *image* (or intensity) plot, with the largest value being white and the smallest value being black.

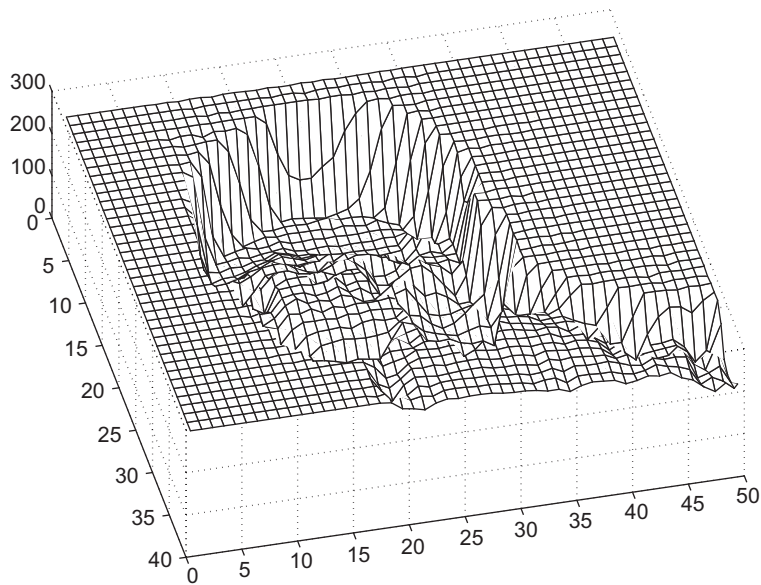
Separable Signals

A separable signal (sequence) satisfies the equation

$$x(n_1, n_2) = x_1(n_1)x_2(n_2) \quad \text{for all } n_1 \text{ and } n_2,$$

**FIGURE 1.1-5**

A contour plot of Eric.

**FIGURE 1.1-6**

A mesh plot of Eric.

**FIGURE 1.1–7**

An intensity plot of Eric.

for some 1-D signals $x_1(n_1)$ and $x_2(n_2)$. If we think of the finite support case, where $x(n_1, n_2)$ can be represented by a matrix, then $x_1(n_1)$ and $x_2(n_2)$ can be represented as column and row vectors, respectively. So, we see that separability is the same as saying that the matrix X can be written as the outer product $X = \mathbf{x}_1 \mathbf{x}_2^T$, which is the same as saying that the matrix X has only one singular value in its *singular value decomposition* (SVD).¹ Clearly, this is very special. Note that while an $N \times N$ matrix X has N^2 degrees of freedom (number of variables), the outer product $\mathbf{x}_1 \mathbf{x}_2^T$ has only $2N$ degrees of freedom. Nevertheless, separable signals play an important role in multidimensional signal processing (MDSP) as representation bases (e.g., Fourier transform) and simple filter impulse responses. A “real” image, like Eric in Figure 1.1–7, regarded as a 100×76 matrix would have extremely many terms in its SVD, i.e., highly nonseparable.

Periodic Signals

A 2-D sequence $x(n_1, n_2)$ is *periodic* with period (N_1, N_2) , also written as $N_1 \times N_2$, if the following equalities hold for all integers n_1 and n_2 :

$$\begin{aligned} x(n_1, n_2) &= x(n_1 + N_1, n_2), \\ &= x(n_1, n_2 + N_2), \end{aligned}$$

where N_1 and N_2 are positive integers.

¹The SVD is a representation of a matrix in terms of its eigenvalues and eigenvectors and is written for a real square matrix as $X = \sum \lambda_i \mathbf{e}_i \mathbf{e}_i^T$, where the λ_i are the eigenvalues and the \mathbf{e}_i are the eigenvectors of X .

The period (N_1, N_2) defines a 2-D grid (either spatial or space-time) over which the signal repeats or is periodic. Since we are in discrete space, the period must be composed of *positive integers*. This type of periodicity occurs often for 2-D signals and is referred to as *rectangular periodicity*. We call the resulting period the *rectangular period*.

Example 1.1–1: A Periodic Signal

An example is the signal $\sin(2\pi n_1/8 + 2\pi n_2/16)$, for which the rectangular period is easily seen to be $(8, 16)$. A separable signal with the same period is $\sin(2\pi n_1/8)\sin(2\pi n_2/16)$. If, however, we remove the factor 2π in the arguments, then we get $\sin(n_1/8 + n_2/16)$ and $\sin(n_1/8)\sin(n_2/16)$, neither of which is periodic at all! This is because we only allow integer values n_1 and n_2 in discrete-parameter signal space.

Given a periodic function, the period effectively defines a *basic cell* in the plane, which can be repeated to form the function over all integers n_1 and n_2 . As such, we often want the minimum size unit cell for efficiency of both specification and storage. In the case of the rectangular period, we seek the smallest nonzero integers that will suffice for N_1 and N_2 to form this basic cell.

Example 1.1–2: Horizontal Wave

Consider the sine wave

$$x(n_1, n_2) = \sin(2\pi n_1/4).$$

The horizontal period is $N_1 = 4$. In the vertical direction, the signal is constant though. So we can use any positive integer N_2 , and we choose the smallest such value, $N_2 = 1$. Thus the rectangular period is $N_1 \times N_2 = 4 \times 1$, and the basic cell consists of the set of points $\{(n_1, n_2) = [(0, 0), (1, 0), (2, 0), (3, 0)]\}$ or any translate of this set.

General Periodicity

There is a more general definition of periodicity that we will encounter from time to time in this text. It is a repetition of blocks, not necessarily rectangular blocks or blocks occurring on a rectangular repeat grid. For the general case, we need to represent the periodicity with two integer vectors, N_1 and N_2 :

$$N_1 = \begin{bmatrix} N_{11} \\ N_{21} \end{bmatrix} \text{ and } N_2 = \begin{bmatrix} N_{12} \\ N_{22} \end{bmatrix}.$$

Then we can write that the 2-D signal $x(n_1, n_2)$ is *general periodic* with period $(N_1, N_2) \triangleq \mathbf{N}$ if the following equalities hold for all integers n_1 and n_2 :

$$\begin{aligned} x(n_1, n_2) &= x(n_1 + N_{11}, n_2 + N_{21}) \\ &= x(n_1 + N_{12}, n_2 + N_{22}). \end{aligned}$$

To avoid degenerate cases, we restrict the integers N_{ij} with the condition

$$\det(\mathbf{N}_1, \mathbf{N}_2) \neq 0.$$

The matrix \mathbf{N} is called the *periodicity matrix*. In matrix notation, the corresponding periodic signal satisfies

$$x(\mathbf{n}) = x(\mathbf{n} + \mathbf{N}\mathbf{r})$$

for all integer vectors $\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$. In words, we can say that the signal repeats itself at all multiples of the shift vectors \mathbf{N}_1 and \mathbf{N}_2 .

Example 1.1–3: General Periodic Signal

An example is the signal $\sin[2\pi(n_1/8 + n_2/16)]$, which is constant along the line $2n_1 + n_2 = 16$. We can compute shift vectors $\mathbf{N}_1 = \begin{pmatrix} 4 \\ 8 \end{pmatrix}$ and $\mathbf{N}_2 = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$. ■

We note that the special case of rectangular periodicity occurs when the periodicity matrix \mathbf{N} is diagonal, for then

$$\mathbf{N} = \begin{bmatrix} N_1 & 0 \\ 0 & N_2 \end{bmatrix},$$

and the rectangular period is (N_1, N_2) , as above. Also in this case, the two period vectors $\mathbf{N}_1 = \begin{pmatrix} N_1 \\ 0 \end{pmatrix}$ and $\mathbf{N}_2 = \begin{pmatrix} 0 \\ N_2 \end{pmatrix}$ lie along the horizontal and vertical axes, respectively.

Example 1.1–4: Cosine Wave

Consider the signal $g(n_1, n_2) = \cos 2\pi(f_1 n_1 + f_2 n_2) = \cos 2\pi \mathbf{f}^T \mathbf{n}$. In continuous space, this signal is certainly periodic, and the rectangular period would be $N_1 \times N_2$, with period vectors $\mathbf{N}_1 = \begin{pmatrix} f_1^{-1} \\ 0 \end{pmatrix}$ and $\mathbf{N}_2 = \begin{pmatrix} 0 \\ f_2^{-1} \end{pmatrix}$. However, this is not a correct answer in discrete space unless the resulting values f_1^{-1} and f_2^{-1} are integers. Generally, if f_1 and f_2 are rational numbers, we can get an integer period as follows: $N_i = p_i f_i^{-1}$, where $f_i = p_i/q_i$,

$i = 1, 2$. If either of the f_i are not rational, there will be no exact rectangular period for this cosine wave. Regarding general periodicity, we are tempted to look for repeats in the direction of the vector $(f_1, f_2)^T$ since $\mathbf{f}^T \mathbf{n}$ is maximized if we increment the vector \mathbf{n} in this direction. However, again we have the problem that this vector would typically not have integer components. We are left with the conclusion that common cosine and sine waves are generally not periodic in discrete space, at least not exactly periodic. The analogous result is also true, although not widely appreciated, in the 1-D case. ■

2-D Discrete-Space Systems

As illustrated in Figure 1.1–8, a 2-D system is defined as a general mathematical operator T that maps each input signal $x(n_1, n_2)$ into a unique output signal $y(n_1, n_2)$.

In equations we write

$$y(n_1, n_2) = T[x(n_1, n_2)],$$

where we remind the signals are assumed to be defined over the entire 2-D discrete space $(-\infty, +\infty) \times (-\infty, +\infty)$, unless otherwise indicated. There is only one restriction on the general system operator T : it must provide a *unique mapping*; that is, for each *input sequence* x there is only one *output sequence* y . Of course, two input sequences could agree only over some area of the plane, but differ elsewhere. Then there can be different output y 's corresponding to these two inputs, since these two inputs are not equal everywhere. In mathematics, an operator such as T is just the generalization of the concept of function, where the input and output spaces are now sequences instead of numbers. The operator T may have an inverse or not. We say that T is invertible if to each output sequence y there corresponds only one input sequence x (i.e., the output determines the input). We denote the inverse operator, if it exists, by T^{-1} .

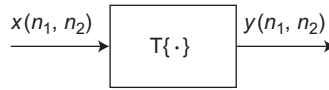


FIGURE 1.1–8

A general 2-D discrete-space system.

Example 1.1–5: Some Simple Systems

Consider the following:

- (a) $y(n_1, n_2) = T[x(n_1, n_2)] = 2x(n_1, n_2)$
- (b) $y(n_1, n_2) = T[x(n_1, n_2)] = x^2(n_1, n_2) + 3x(n_1, n_2) + 5$
- (c) $y(n_1, n_2) = T[x(n_1, n_2)] = 3x(n_1, n_2) + 5$
- (d) $y(n_1, n_2) = T[x(n_1, n_2)] = \frac{1}{2}[x(n_1, n_2) + x(n_1 - 1, n_2)]$

We note by inspection that the 2-D system operators given in items (a) and (c) are invertible, while that given in item (b) is not. The system given by item (d) has *memory* while those in items (a)–(c) do not. If we call (n_1, n_2) the *present*, then a *memoryless system* is one whose present output only depends on present inputs. ■

A special case of the general 2-D system is the linear system (Definition 1.1–1).

Definition 1.1–1: Linear System

A 2-D discrete system is *linear* if the following equation holds for all input-output sequence pairs $x_i - y_i$ and all scaling constants a_i :

$$L[a_1x_1(n_1, n_2) + a_2x_2(n_1, n_2)] = a_1L[x_1(n_1, n_2)] + a_2L[x_2(n_1, n_2)],$$

where we have denoted the linear operator by L . ■

As in 1-D signal processing, linear systems are very important to the theory of 2-D signal processing. The reasons are the same: (1) we know the most about linear systems, (2) approximate linear systems arise a lot in practice, and (3) many adaptive nonlinear systems are composed of linear blocks, designed using linear system theory. In the previous example, systems (a) and (d) are linear by this definition. A simple necessary condition for linearity is that the output doubles when the input doubles, and this rules out systems (b) and (c).

Definition 1.1–2: Shift Invariance

A system T is *shift-invariant* if for an arbitrary input x , any finite shift of x produces the identical shift in the corresponding output y . That is, if $T[x(n_1, n_2)] = y(n_1, n_2)$, then for all (integer) shifts (m_1, m_2) we have $T[x(n_1 - m_1, n_2 - m_2)] = y(n_1 - m_1, n_2 - m_2)$. ■

Often we think in terms of a *shift vector*, denoted $\mathbf{m} = (m_1, m_2)^T$. In fact, we can just as well write the preceding two definitions in the more compact vector notation as

$$\text{linearity: } L[a_1x_1(\mathbf{n}) + a_2x_2(\mathbf{n})] = a_1L[x_1(\mathbf{n})] + a_2L[x_2(\mathbf{n})]$$

$$\text{shift invariance: } T[x(\mathbf{n} - \mathbf{m})] = y(\mathbf{n} - \mathbf{m}) \text{ for all integer shift vectors } \mathbf{m}$$

The vector notation is very useful for 3-D systems, such as those occurring in video signal processing (see Chapter 11).

2-D Convolution

Shift invariant linear systems can be represented by convolution. If a system is *linear shift-invariant* (LSI), then we can write, using the shifting representation (1.1–1),

$$y(n_1, n_2) = L \left[\sum_{k_1, k_2} x(k_1, k_2) \delta(n_1 - k_1, n_2 - k_2) \right]$$

$$\begin{aligned}
&= \sum_{k_1, k_2} x(k_1, k_2) L[\delta(n_1 - k_1, n_2 - k_2)] \\
&= \sum_{k_1, k_2} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2),
\end{aligned}$$

where the sequence h is called the LSI system's *impulse response*, defined as $h(n_1, n_2) \triangleq L[\delta(n_1, n_2)]$. We define the 2-D convolution operator $*$ as

$$(x * h)(n_1, n_2) \triangleq \sum_{k_1, k_2} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2). \quad (1.1-2)$$

It then follows that for a 2-D LSI system we have

$$\begin{aligned}
y(n_1, n_2) &= (x * h)(n_1, n_2) \\
&= (h * x)(n_1, n_2),
\end{aligned}$$

where the latter equality follows easily by substitution in (1.1-2). Again, we remind that all the summations over k_1, k_2 range from $-\infty$ to $+\infty$.

Properties of 2-D Convolution or Convolution Algebra

1. Commutativity: $x * y = y * x$
2. Associativity: $(x * y) * z = x * (y * z)$
3. Distributivity: $x * (y + z) = x * y + x * z$
4. Identity element: $\delta(n_1, n_2)$ with property $\delta * x = x$
5. Zero element: $0(n_1, n_2) = 0$ with property $0 * x = 0$ ²

All of these properties of convolution hold for any 2-D signals x , y , and z , for which convolution is defined (i.e., for which the infinite sums exist).

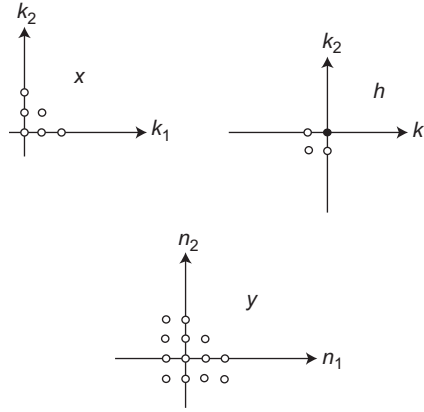
Example 1.1-6: Spatial Convolution

In Figure 1.1-9, we see an example of 2-D or spatial convolution. The impulse response h has been reversed or reflected through the origin in both parameters to yield $h(n_1 - k_1, n_2 - k_2)$, shown in this $k_1 \times k_2$ plane figure for $n_1 = n_2 = 0$. This sequence is then shifted around the $k_1 \times k_2$ plane via integer vector shifts (n_1, n_2) , and then a sum of products is taken with input $x(k_1, k_2)$ to give output $y(n_1, n_2)$.

In general, for signals with rectangular support, we have the following result for the support of their convolution:

$$\begin{aligned}
&\text{If } \text{supp}(x) = N_1 \times N_2 \text{ and } \text{supp}(h) = M_1 \times M_2, \\
&\text{then } \text{supp}(y) = (N_1 + M_1 - 1) \times (N_2 + M_2 - 1).
\end{aligned}$$

²Note that 0 here is the zero sequence defined as $0(n_1, n_2) = 0$ for all (n_1, n_2) .

**FIGURE 1.1-9**

An example of 2-D or spatial convolution.

For signals of non rectangular support, this result can be used to rectangularly bound the support of the output.

Stability in 2-D Systems

Stable systems are those for which a small change in the input gives a small change in the output. As such, they are very useful in applications. We can mathematically define *bounded-input bounded-output* (BIBO) stability for 2-D systems analogously to that in 1-D system theory. A spatial or 2-D system will be *stable* if the response to every *uniformly* bounded input is itself uniformly bounded. It is generally very difficult to verify this condition, but for an LSI system the condition is equivalent to the impulse response being absolutely summable; that is,

$$\sum_{k_1, k_2} |h(k_1, k_2)| < \infty. \quad (1.1-3)$$

Theorem 1.1-1: LSI Stability

A 2-D LSI system is BIBO stable if and only if its impulse response $h(n_1, n_2)$ is absolutely summable.

Proof We start by assuming that (1.1-3) is true. Then, by the convolution representation, we have

$$|y(n_1, n_2)| = \left| \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) \right|$$

$$\begin{aligned}
&\leq \sum_{k_1, k_2} |h(k_1, k_2)| |x(n_1 - k_1, n_2 - k_2)| \\
&\leq \left(\sum_{k_1, k_2} |h(k_1, k_2)| \right) \max |x(n_1, n_2)| < \infty,
\end{aligned}$$

for any uniformly bounded input x , thus establishing *sufficiency* of (1.1–3). To show necessity, following Oppenheim and Schaffer [1] in the one-dimensional case, we can choose the uniformly bounded input signal $x(n_1, n_2) = \exp -j\theta(-n_1, -n_2)$, where $\theta(n_1, n_2)$ is the argument function of the complex function $h(n_1, n_2)$; then the system output at $(n_1, n_2) = (0, 0)$ will be given by (1.1–3), thus showing that absolute summability of the impulse response is also *necessary* for a bounded output. Having shown both sufficiency and necessity, we are done. ■

In mathematics, the *function space* of absolutely summable 2-D sequences is often denoted l^1 , so we can also write $h \in l^1$ as the condition for an LSI system with impulse response h to be BIBO stable. Thus, this well-known 1-D mathematical result [1] easily generalizes to the 2-D case.

1.2 2-D DISCRETE-SPACE FOURIER TRANSFORM

The Fourier transform is important in 1-D signal processing because it effectively explains the operation of linear time-invariant (LTI) systems via the concept of frequency response. This frequency response is just the Fourier transform of the system impulse response. While convolution provides a complicated description of the LTI system operation, where generally the input at all locations n affects the output at all locations, the frequency response provides a simple interpretation as a scalar weighting in the Fourier domain, where the output at each frequency ω depends only on the input at that same frequency. A similar result holds for 2-D systems that are LSI, as we show in the following discussions. ■

Definition 1.2–1: 2-D Fourier Transform

We define the *2-D Fourier transform* as follows:

$$X(\omega_1, \omega_2) \triangleq \sum_{n_1, n_2}^{+\infty}_{-\infty} x(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2). \quad (1.2-1)$$

The radian frequency variable ω_1 is called *horizontal frequency*, and the variable ω_2 is called *vertical frequency*. The domain of (ω_1, ω_2) is the entire plane $(-\infty, +\infty) \times (-\infty, +\infty) \triangleq (-\infty, +\infty)^2$. ■

One of the easy properties of the 2-D Fourier transform is that it is periodic with rectangular period $2\pi \times 2\pi$, a property originating from the integer argument values of n_1 and n_2 . To see this property simply note

$$\begin{aligned}
 X(\omega_1 \pm 2\pi, \omega_2 \pm 2\pi) &= \sum_{\substack{n_1, n_2 \\ -\infty \\ +\infty}} x(n_1, n_2) \exp -j[(\omega_1 \pm 2\pi)n_1 + (\omega_2 \pm 2\pi)n_2] \\
 &= \sum_{\substack{n_1, n_2 \\ -\infty \\ +\infty}} x(n_1, n_2) \exp -j\omega_1 n_1 \exp -j2\pi n_1 \exp -j\omega_2 n_2 \exp -j2\pi n_2 \\
 &= \sum_{\substack{n_1, n_2 \\ -\infty \\ +\infty}} x(n_1, n_2) (\exp -j\omega_1 n_1) \cdot 1 \cdot (\exp -j\omega_2 n_2) \cdot 1 \\
 &= X(\omega_1, \omega_2).
 \end{aligned}$$

Thus the 2-D Fourier transform only needs be calculated for one period, usually taken to be $[-\pi, +\pi] \times [-\pi, +\pi]$. It is analogous to the 1-D case where the Fourier transform $X(\omega)$ has period 2π and is usually just calculated for $[-\pi, +\pi]$. Upon close examination, we can see that the 2-D Fourier transform is closely related to the 1-D Fourier transform. In fact, we can rewrite (1.2-1) as

$$\begin{aligned}
 X(\omega_1, \omega_2) &= \sum_{n_1} \sum_{n_2} x(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_2} \sum_{n_1} x(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_2} \left[\sum_{n_1} x(n_1, n_2) \exp -j\omega_1 n_1 \right] \exp -j\omega_2 n_2 \\
 &= \sum_{n_2} X(\omega_1; n_2) \exp -j\omega_2 n_2,
 \end{aligned}$$

where $X(\omega_1; n_2) \triangleq \sum_{n_1} x(n_1, n_2) \exp -j\omega_1 n_1$, the Fourier transform of row n_2 . In words, we say that the 2-D Fourier transform can be decomposed as a set of 1-D Fourier transforms on all the rows of x , followed by another set of 1-D Fourier transforms on the columns $X(\omega_1; n_2)$ that result from the first set of transforms. We call the 2-D Fourier transform a *separable operator*, because it can be performed as the concatenation of 1-D operations on the rows followed by 1-D operations on the columns, or *vice versa*. Such 2-D operators are common in MDSP and offer great computational simplification when they occur.

The Fourier transform has been studied for convergence of the infinite sum in several senses. First, if we have an absolutely sumable signal, then the Fourier transform sum will converge in the sense of *uniform convergence*; as a result, it will be a continuous function of ω_1 and ω_2 . A weaker form of convergence of the Fourier transform sum is *mean-square convergence* [2]. This is the type of convergence that leads to a Fourier transform that is a discontinuous function. A third type of convergence is as a *generalized function*, such as $\delta(\omega_1, \omega_2)$, the 2-D Dirac delta function. It is used for periodic signals such as $\cos 2\pi(\alpha_1 n_1 + \alpha_2 n_2)$, whose Fourier transform will be shown to be the scaled and shifted 2-D Dirac impulse $(2\pi)^2 \delta(\omega_1 - \alpha_1, \omega_2 - \alpha_2)$ when $|\alpha_1| < \pi$ and $|\alpha_2| < \pi$, and when attention is limited to the unit cell $[-\pi, +\pi] \times [-\pi, +\pi]$. Outside this cell, the function must repeat.

In operator notation, we can write the Fourier transform relation as

$$X = \text{FT}[x],$$

indicating that the Fourier transform operator FT maps 2-D sequences into 2-D functions X that happen to be continuous-parameter and periodic functions with period $2\pi \times 2\pi$.

Example 1.2-1: Fourier Transform of Rectangular Pulse Function

Let

$$x(n_1, n_2) = \begin{cases} 1 & 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1 \\ 0 & \text{else.} \end{cases}$$

Then its 2-D Fourier transform can be determined as

$$\begin{aligned} X(\omega_1, \omega_2) &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 1 \exp -j(\omega_1 n_1 + \omega_2 n_2) \\ &= \left(\sum_{n_1=0}^{N_1-1} \exp -j\omega_1 n_1 \right) \left(\sum_{n_2=0}^{N_2-1} \exp -j\omega_2 n_2 \right) \\ &= \left(\frac{1 - e^{-j\omega_1 N_1}}{1 - e^{-j\omega_1}} \right) \left(\frac{1 - e^{-j\omega_2 N_2}}{1 - e^{-j\omega_2}} \right) \\ &= e^{-j\omega_1(N_1-1)/2} \frac{\sin \omega_1 N_1/2}{\sin \omega_1/2} e^{-j\omega_2(N_2-1)/2} \frac{\sin \omega_2 N_2/2}{\sin \omega_2/2} \\ &= e^{-j[\omega_1(N_1-1)/2 + \omega_2(N_2-1)/2]} \frac{\sin \omega_1 N_1/2}{\sin \omega_1/2} \frac{\sin \omega_2 N_2/2}{\sin \omega_2/2}, \end{aligned}$$

where the second to last line follows by factoring out the indicated phase term from each of the two factors in the line above, and then using Euler's equality $\sin \theta = \frac{1}{2j}(e^{j\theta} - e^{-j\theta})$ in top and bottom terms of each factor. We see the result is just the product of two

1-D Fourier transforms for this separable and rectangular 2-D pulse function. If N_1 and N_2 are odd numbers, we can shift the pulse to be centered on the origin $(0,0)$ and thereby remove the linear phase shift term out in front, corresponding to a shift (delay) of $\left(\frac{1}{2}(N_1 - 1), \frac{1}{2}(N_2 - 1)\right)$. However, because of the way we have written the signal x as starting at $(0,0)$, there remains this linear phase shift. A contour plot of the log magnitude of this function is provided in Figure 1.2-1, and the corresponding 3-D perspective plot is shown in Figure 1.2-2. Both of these plots were produced with MATLAB. ■

Example 1.2-2: Fourier Transform of Line Impulse

Consider a line impulse of the form

$$x(n_1, n_2) = \delta(n_1 - n_2) = \begin{cases} 1, & n_2 = n_1, \\ 0, & \text{else,} \end{cases}$$

which is a line of slope 1 in the $n_1 \times n_2$ plane or an angle of 45° . We can take the Fourier transform as

$$\begin{aligned} X(\omega_1, \omega_2) &= \sum_{n_1, n_2} \delta(n_1 - n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\ &= \sum_{n_1=-\infty}^{+\infty} \exp -j\omega_1 n_1 \exp -j\omega_2 n_1 \\ &= \sum_{n_1=-\infty}^{+\infty} \exp -j(\omega_1 + \omega_2) n_1 \\ &= 2\pi \delta(\omega_1 + \omega_2) \quad \text{for } (\omega_1, \omega_2) \in [-\pi, +\pi]^2, \end{aligned} \tag{1.2-2} \tag{1.2-3}$$

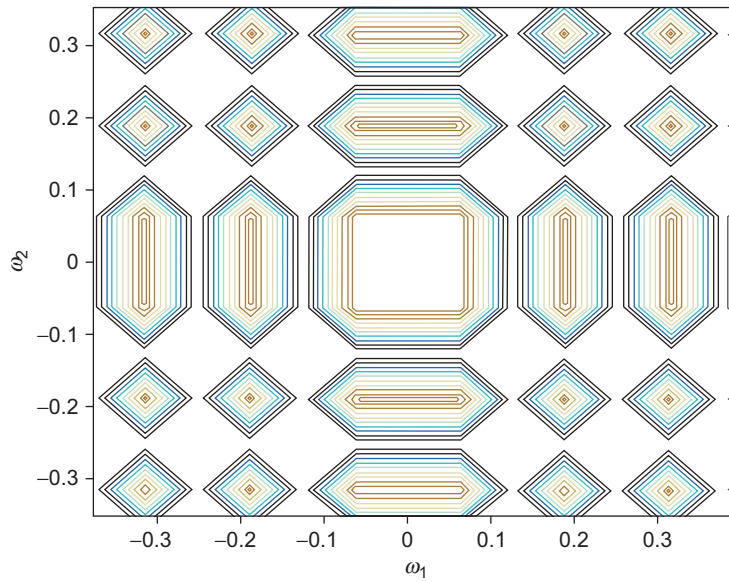
which is a Dirac impulse line in the 2-D frequency domain, along the line $\omega_2 = -\omega_1$. Thus the Fourier transform of an impulse line is a Dirac impulse line in the 2-D frequency domain. Note that the angle of this line is that of the spatial domain line plus/minus 90° ; that is, the two lines are perpendicular to one another. Why is this result reasonable? ■

Inverse 2-D Fourier Transform

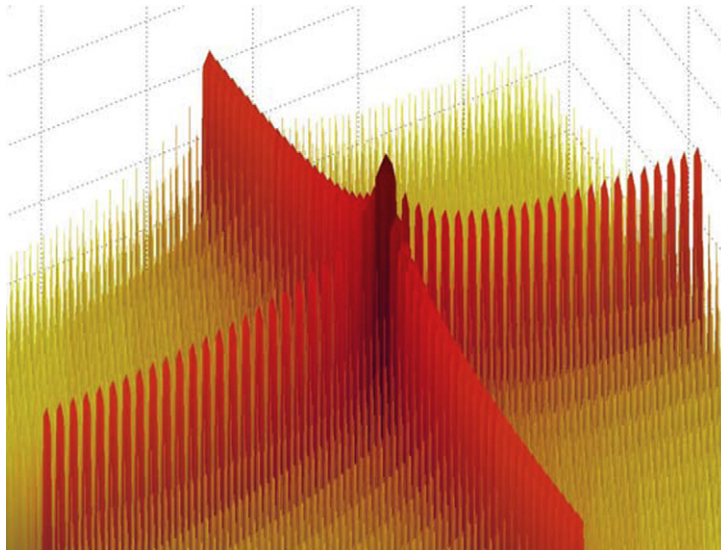
The inverse 2-D Fourier transform is given as

$$x(n_1, n_2) = \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} X(\omega_1, \omega_2) \exp +j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2. \tag{1.2-4}$$

Note that we only integrate over a “unit” cell $[-\pi, +\pi] \times [-\pi, +\pi]$ in the $\omega_1 \times \omega_2$ frequency domain. To justify this formula, we can plug in (1.2-1) and interchange

**FIGURE 1.2-1**

Zoomed-in contour plot of log magnitude of 2-D rectangular sinc function with $N_1 = N_2 = 50$.

**FIGURE 1.2-2**

3-D perspective plot of log magnitude of 2-D rectangular sinc function.

the order of summation and integration, just as done in the 1-D case. Alternatively, we can use the fact that the 2-D FT is a separable operator and just use the known results for transform and inverse in the 1-D case to arrive at this same answer for the inverse 2-D Fourier transform. In operator notation, we denote the inverse Fourier transform as IFT and write

$$x = \text{IFT}[X].$$

A 2-D proof of this result follows closely the 1-D result [1]. First, we insert (1.2-1) into (1.2-4) to obtain

$$\begin{aligned} x(n_1, n_2) &= \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} \left[\sum_{\substack{l_1, l_2 \\ -\infty \\ +\infty}} x(l_1, l_2) \exp -j(\omega_1 l_1 + \omega_2 l_2) \right] \\ &\quad \times \exp +j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2 \\ &= \sum_{l_1, l_2} x(l_1, l_2) \left\{ \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} \right. \\ &\quad \times \exp +j[\omega_1(n_1 - l_1) + \omega_2(n_2 - l_2)] d\omega_1 d\omega_2 \left. \right\} \\ &= \sum_{l_1, l_2} x(l_1, l_2) \left\{ \left[\frac{1}{2\pi} \int_{[-\pi, +\pi]} \exp +j\omega_1(n_1 - l_1) d\omega_1 \right] \right. \\ &\quad \times \left. \left[\frac{1}{2\pi} \int_{[-\pi, +\pi]} \exp +j\omega_2(n_2 - l_2) d\omega_2 \right] \right\}. \end{aligned}$$

Now,

$$\left(\frac{1}{2\pi} \int_{[-\pi, +\pi]} \exp +j\omega m d\omega \right) = \frac{\exp +j\omega m}{j2\pi m} \Big|_{-\pi}^{+\pi} = \frac{\sin \pi m}{\pi m} = \delta(m),$$

so substituting this result in the above equation yields

$$\begin{aligned} x(n_1, n_2) &= \sum_{l_1, l_2} x(l_1, l_2) \delta(l_1) \delta(l_2) \\ &= x(n_1, n_2) \end{aligned}$$

as was to be shown.

A main application of 2-D Fourier transforms is to provide a simplified view of spatial convolution as used in linear filtering, our next topic.

Fourier Transform of 2-D or Spatial Convolution

Theorem 1.2–1: Fourier Convolution Theorem

If

$$y(n_1, n_2) = h(n_1, n_2) * x(n_1, n_2),$$

then

$$Y(\omega_1, \omega_2) = H(\omega_1, \omega_2)X(\omega_1, \omega_2).$$

Proof

$$\begin{aligned}
 Y(\omega_1, \omega_2) &= \sum_{\substack{n_1, n_2 \\ -\infty \\ +\infty}} y(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_1, n_2} \left[\sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) \right] \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_1, n_2} \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) \\
 &\quad \times \exp -j\{\omega_1 [k_1 + (n_1 - k_1)] + \omega_2 [k_2 + (n_2 - k_2)]\} \\
 &= \sum_{n_1, n_2} \sum_{k_1, k_2} h(k_1, k_2) \exp -j(\omega_1 k_1 + \omega_2 k_2) x(n_1 - k_1, n_2 - k_2) \\
 &\quad \times \exp -j[\omega_1 (n_1 - k_1) + \omega_2 (n_2 - k_2)] \\
 &= \sum_{k_1, k_2} h(k_1, k_2) \exp -j(\omega_1 k_1 + \omega_2 k_2) \\
 &\quad \times \left\{ \sum_{n_1, n_2} x(n_1 - k_1, n_2 - k_2) \exp -j[\omega_1 (n_1 - k_1) + \omega_2 (n_2 - k_2)] \right\} \\
 &= \sum_{k_1, k_2} h(k_1, k_2) \exp -j(\omega_1 k_1 + \omega_2 k_2) \left[\sum_{n'_1, n'_2} x(n'_1, n'_2) \exp -j(\omega_1 n'_1 + \omega_2 n'_2) \right],
 \end{aligned}$$

with $n'_1 \triangleq n_1 - k_1$ and $n'_2 \triangleq n_2 - k_2$.

Now, since the limits of the sums are infinite, the shift by any finite value (k_1, k_2) makes no difference, and the limits on the inside sum over (n'_1, n'_2) remain at $(-\infty, +\infty) \times (-\infty, +\infty)$. Thus we can bring its double sum outside the sum over (k_1, k_2) and recognize it as $X(\omega_1, \omega_2)$, the Fourier transform of the input signal x . What is left inside is the frequency response $H(\omega_1, \omega_2)$, and so we have finally

$$\begin{aligned} &= \left[\sum_{k_1, k_2} h(k_1, k_2) \exp -j(\omega_1 k_1 + \omega_2 k_2) \right] \left[\sum_{n'_1, n'_2} x(n'_1, n'_2) \exp -j(\omega_1 n'_1 + \omega_2 n'_2) \right] \\ &= H(\omega_1, \omega_2) \cdot X(\omega_1, \omega_2) \end{aligned}$$

as was to be shown. ■

Since we have seen that a 2-D or spatial LSI system is characterized by its impulse response $h(n_1, n_2)$, we now see that its frequency response $H(\omega_1, \omega_2)$ also suffices to characterize such a system. And the Fourier transform Y of the output equals the product of the frequency response H and the Fourier transform X of the input. When the frequency response H takes on only values 1 and 0, we call the system an *ideal filter*, since such an LSI system will filter out some frequencies and pass others unmodified. More generally, the term filter has grown to include all such LSI systems, and has been extended to shift-variant and even nonlinear systems through the concept of the Volterra series of operators [3].

Example 1.2–3: Fourier Transform of Complex Plane Wave

Let $x(n_1, n_2) = A \exp j(\omega_1^0 n_1 + \omega_2^0 n_2) \triangleq e(n_1, n_2)$, where $|\omega_i^0| < \pi$; then $E(\omega_1, \omega_2) = c \delta(\omega_1 - \omega_1^0, \omega_2 - \omega_2^0)$ in the basic square $[-\pi, +\pi]^2$. Finding the constant c by inverse Fourier transform, we conclude that $c = (2\pi)^2 A$. Inputting this plane wave into an LSI system or filter with frequency response $H(\omega_1, \omega_2)$, we obtain the Fourier transform output signal $Y(\omega_1, \omega_2) = (2\pi)^2 A H(\omega_1^0, \omega_2^0) \delta(\omega_1 - \omega_1^0, \omega_2 - \omega_2^0) = H(\omega_1^0, \omega_2^0) E(\omega_1^0, \omega_2^0)$, or in the space domain $y(n_1, n_2) = H(\omega_1^0, \omega_2^0) e(n_1, n_2)$, thus showing that complex exponentials (i.e., plane waves) are the *eigenfunctions* of spatial LSI systems, and the frequency response H evaluated at the plane wave frequency (ω_1^0, ω_2^0) becomes the corresponding *eigenvalue*. ■

Some Important Properties of the FT Operator

1. Linearity: $ax + by \Leftrightarrow aX(\omega_1, \omega_2) + bY(\omega_1, \omega_2)$
2. Convolution: $x * y \Leftrightarrow X(\omega_1, \omega_2)Y(\omega_1, \omega_2)$
3. Multiplication: $xy \Leftrightarrow (X * Y)(\omega_1, \omega_2) \triangleq \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} X(v_1, v_2) Y(\omega_1 - v_1, \omega_2 - v_2) dv_1 dv_2$
4. Modulation: $x(n_1, n_2) \exp j(v_1 n_1 + v_2 n_2) \Leftrightarrow X(\omega_1 - v_1, \omega_2 - v_2)$ for integers v_1 and v_2
5. Shift (delay): $x(n_1 - m_1, n_2 - m_2) \Leftrightarrow X(\omega_1, \omega_2) \exp -j(\omega_1 m_1 + \omega_2 m_2)$

6. Partial differentiation in frequency domain:

$$\begin{aligned} -jn_1x(n_1, n_2) &\Leftrightarrow \frac{\partial X}{\partial \omega_1} \\ -jn_2x(n_1, n_2) &\Leftrightarrow \frac{\partial X}{\partial \omega_2} \end{aligned}$$

7. “Initial” value:

$$x(0, 0) = \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} X(\omega_1, \omega_2) d\omega_1 d\omega_2$$

8. “DC” value:

$$X(0, 0) = \sum_{\substack{n_1, n_2 \\ -\infty \\ +\infty}} x(n_1, n_2)$$

9. Parseval’s theorem:

$$\sum_{n_1, n_2} x(n_1, n_2) y^*(n_1, n_2) = \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} X(\omega_1, \omega_2) Y^*(\omega_1, \omega_2) d\omega_1 d\omega_2$$

with the “power” version, obtained by letting $y = x$

$$\sum_{n_1, n_2} |x(n_1, n_2)|^2 = \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} |X(\omega_1, \omega_2)|^2 d\omega_1 d\omega_2$$

10. Separable signal:

$$x_1(n_1)x_2(n_2) \Leftrightarrow X_1(\omega_1)X_2(\omega_2)$$

Some Useful Fourier Transform Pairs

1. Constant c :

$$\text{FT}\{c\} = (2\pi)^2 c \delta(\omega_1, \omega_2)$$

in the unit cell $[-\pi, +\pi]^2$

2. Complex exponential for spatial frequency $(\nu_1, \nu_2) \in [-\pi, +\pi]^2$:

$$\text{FT}\{\exp j(\nu_1 n_1 + \nu_2 n_2)\} = (2\pi)^2 \delta(\omega_1 - \nu_1, \omega_2 - \nu_2)$$

in the unit cell $[-\pi, +\pi]^2$

3. Constant in n_2 dimension:

$$\text{FT}\{x_1(n_1)\} = 2\pi X_1(\omega_1) \delta(\omega_2),$$

where $X_1(\omega_1)$ is a 1-D Fourier transform and $\delta(\omega_2)$ is a 1-D impulse function

4. Ideal lowpass filter (rectangular passband):

$$H_r(\omega_1, \omega_2) = I_{\omega_c}(\omega_1) I_{\omega_c}(\omega_2), \quad \text{where } I_{\omega_c} \triangleq \begin{cases} 1 & |\omega| \leq \omega_c, \\ 0 & \text{else,} \end{cases}$$

with ω_c the *cutoff frequency* of the filter. Necessarily $\omega_c \leq \pi$. The function I_{ω_c} is sometimes called an *indicator function*, since it indicates the passband. Taking the inverse 2-D Fourier transform of this separable function, we proceed as follows to obtain the ideal impulse response:

$$\begin{aligned}
 h_r(n_1, n_2) &= \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} H_r(\omega_1, \omega_2) \exp + j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2 \\
 &= \left[\frac{1}{2\pi} \int_{[-\pi, +\pi]} I_{\omega_c}(\omega_1) \exp + j\omega_1 n_1 d\omega_1 \right] \\
 &\quad \times \left[\frac{1}{2\pi} \int_{[-\pi, +\pi]} I_{\omega_c}(\omega_2) \exp + j\omega_2 n_2 d\omega_2 \right] \\
 &= \frac{\sin \omega_c n_1}{\pi n_1} \cdot \frac{\sin \omega_c n_2}{\pi n_2}, \quad -\infty < n_1, n_2 < +\infty.
 \end{aligned}$$

5. Ideal lowpass filter (circular passband) $\omega_c < \pi$

$$H_c(\omega_1, \omega_2) = \begin{cases} 1 & \sqrt{\omega_1^2 + \omega_2^2} \leq \omega_c, \\ 0 & \text{else,} \end{cases} \quad \text{for } (\omega_1, \omega_2) \in [-\pi, +\pi] \times [-\pi, +\pi].$$

The inverse Fourier transform of this circular symmetric frequency response can be represented as the integral

$$\begin{aligned}
 h_c(n_1, n_2) &= \frac{1}{(2\pi)^2} \int \int_{\sqrt{\omega_1^2 + \omega_2^2} \leq \omega_c} 1 \exp + j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2 \\
 &= \frac{1}{(2\pi)^2} \int_0^{\omega_c} \int_{-\pi}^{+\pi} \exp + ju(n_1 \cos \theta + n_2 \sin \theta) u du d\theta \\
 &= \frac{1}{(2\pi)^2} \int_0^{\omega_c} u \left\{ \int_{-\pi}^{+\pi} \exp + j[ur \cos(\theta - \phi)] d\theta \right\} du \\
 &= \frac{1}{(2\pi)^2} \int_0^{\omega_c} u \left[\int_{-\pi}^{+\pi} \exp + j(ur \cos \theta) d\theta \right] du,
 \end{aligned}$$

where we have used, first, polar coordinates in frequency, $\omega_1 = u \cos \theta$ and $\omega_2 = u \sin \theta$, and then in the next line, polar coordinates in space, $n_1 = r \cos \phi$ and $n_2 = r \sin \phi$. Finally, the last line follows because the integral over θ does not depend on ϕ since it is an integral over the full period 2π . The inner integral over θ can now be

recognized in terms of the zeroth-order Bessel function of the first kind $J_0(x)$, with integral representation [4, 5]

$$J_0(x) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \cos(x \cos \theta) d\theta = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \cos(x \sin \theta) d\theta.$$

To see this, we note the integral

$$\int_{-\pi}^{+\pi} \exp + j(ur \cos \theta) d\theta = \int_{-\pi}^{+\pi} \cos(ur \cos \theta) d\theta,$$

since the imaginary part, via Euler's formula, is an odd function integrated over even limits, and hence is zero. So, continuing, we can write

$$\begin{aligned} h_c(n_1, n_2) &= \frac{1}{(2\pi)^2} \int_0^{\omega_c} u \left[\int_{-\pi}^{+\pi} \exp + j(ur \cos \theta) d\theta \right] du \\ &= \frac{1}{2\pi} \int_0^{\omega_c} u J_0(ur) du \\ &= \frac{\omega_c}{2\pi r} J_1(\omega_c r) \\ &= \frac{\omega_c}{2\pi \sqrt{n_1^2 + n_2^2}} J_1(\omega_c \sqrt{n_1^2 + n_2^2}), \end{aligned}$$

where J_1 is the first-order Bessel function of the first kind $J_1(x)$, satisfying the known relation [6, p. 484]

$$xJ_1(x) = \int_0^x uJ_0(u) du.$$

Comparing these two ideal lowpass filters, one with a square passband and the other circular with diameter equal to a side of the square, we get the two impulse responses along the n_1 axis:

$$\begin{aligned} h_r(n_1, 0) &= \frac{\omega_c}{\pi} \frac{\sin \omega_c n_1}{\pi n_1} \quad \text{and} \\ h_c(n_1, 0) &= \frac{\omega_c}{2\pi n_1} J_1(\omega_c n_1). \end{aligned}$$

These 1-D sequences are plotted via MATLAB in Figure 1.2–3. Note their similarity.

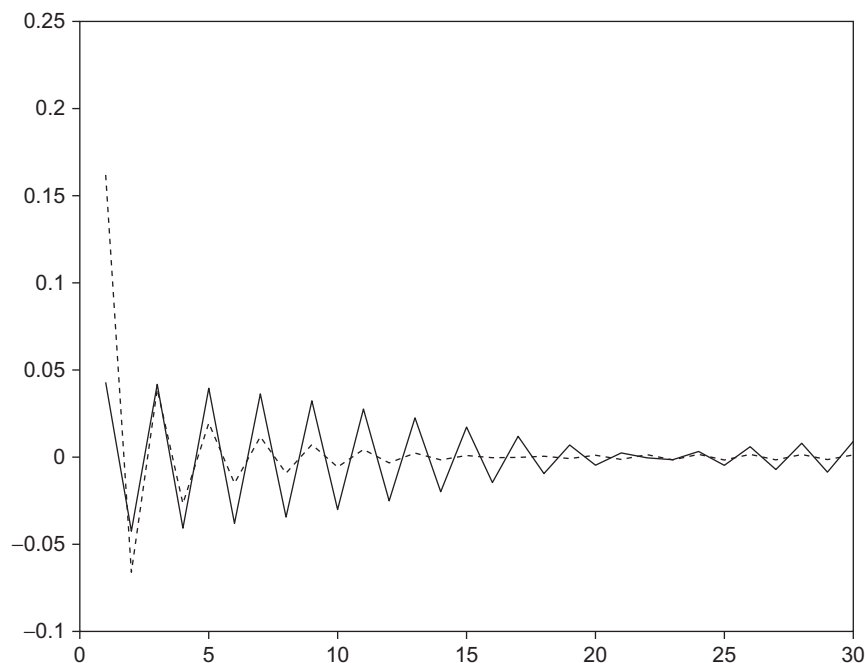


FIGURE 1.2-3

Two impulse responses of 2-D ideal lowpass filters. The solid curve is “rectangular” and the dashed curve is “circular.”

Example 1.2-4: Fourier Transform of Separable Signal

Let $x(n_1, n_2) = x_1(n_1)x_2(n_2)$, then when we compute the Fourier transform, the following simplification arises:

$$\begin{aligned}
 X(\omega_1, \omega_2) &= \sum_{n_1} \sum_{n_2} x(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_1} \sum_{n_2} x_1(n_1) x_2(n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\
 &= \sum_{n_1} x_1(n_1) \left[\sum_{n_2} x_2(n_2) \exp -j\omega_2 n_2 \right] \exp -j\omega_1 n_1 \\
 &= \left[\sum_{n_1} x_1(n_1) \exp -j\omega_1 n_1 \right] \left[\sum_{n_2} x_2(n_2) \exp -j\omega_2 n_2 \right] \\
 &= X_1(\omega_1) X_2(\omega_2).
 \end{aligned}$$

A consequence of this result is that in 2-D signal processing, multiplication in the spatial domain does not always lead to convolution in the frequency domain! Can you reconcile this fact with the basic 2-D Fourier transform property 3? (See problem 9 at the end of this chapter.)

Symmetry Properties of the Fourier Transform

Now we will consider some symmetry properties of the Fourier transform of complex signal $x(n_1, n_2)$. We start with the original FT pair:

$$x(n_1, n_2) \Leftrightarrow X(\omega_1, \omega_2).$$

First, *reflection through the origin* (a generalization of time reversal in the 1-D case): We ask, what is the FT of the signal $x(-n_1, -n_2)$, where each of the axes are reversed? For an image, this corresponds to flipping it right to left, and then flipping it top to bottom. We seek

$$\begin{aligned} \sum_{n_1} \sum_{n_2} x(-n_1, -n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\ = \sum_{n'_1} \sum_{n'_2} x(n'_1, n'_2) \exp -j[\omega_1 (-n'_1) + \omega_2 (-n'_2)] \end{aligned}$$

upon setting $n'_1 = -n_1$ and $n'_2 = -n_2$, so moving the minus signs to the ω_i terms, we have

$$\sum_{n'_1} \sum_{n'_2} x(n'_1, n'_2) \exp -j[(-\omega_1)n'_1 + (-\omega_2)n'_2],$$

which can be seen as $X(-\omega_1, -\omega_2)$. Thus we have the new transform pair

$$x(-n_1, -n_2) \Leftrightarrow X(-\omega_1, -\omega_2).$$

If the original spatial signal is instead conjugated, we have

$$x^*(n_1, n_2) \Leftrightarrow X^*(-\omega_1, -\omega_2), \quad (1.2-5)$$

which generalizes the 1-D FT pair $x^*(n) \Leftrightarrow X^*(-\omega)$. Combining these two 2-D transform pairs, we get the transform of $x^*(-n_1, -n_2)$,

$$x^*(-n_1, -n_2) \Leftrightarrow X^*(\omega_1, \omega_2),$$

resulting in the conjugate of the original $X(\omega_1, \omega_2)$. We can organize these four basic transform pairs using the concept of *conjugate symmetric* and *conjugate antisymmetric* parts, as in one dimension [1].

Definition 1.2-2: Signal Symmetries

The *conjugate symmetric part* of x is

$$x_e(n_1, n_2) \triangleq \frac{1}{2} [x(n_1, n_2) + x^*(-n_1, -n_2)].$$

The *conjugate antisymmetric part* of x is

$$x_o(n_1, n_2) \triangleq \frac{1}{2} [x(n_1, n_2) - x^*(-n_1, -n_2)].$$

From Definition 1.2–2, we get the following two transform pairs:

$$x_e(n_1, n_2) \Leftrightarrow \operatorname{Re} X(\omega_1, \omega_2),$$

$$x_o(n_1, n_2) \Leftrightarrow j \operatorname{Im} X(\omega_1, \omega_2).$$

Similarly, the conjugate symmetric and antisymmetric parts in the 2-D frequency are defined.

Definition 1.2–3:

The *conjugate symmetric part* of X is

$$X_e(\omega_1, \omega_2) \triangleq \frac{1}{2} [X(\omega_1, \omega_2) + X^*(-\omega_1, -\omega_2)].$$

The *conjugate antisymmetric part* of X is

$$X_o(\omega_1, \omega_2) \triangleq \frac{1}{2} [X(\omega_1, \omega_2) - X^*(-\omega_1, -\omega_2)].$$

Using these symmetry properties, we get the following two general transform pairs:

$$\operatorname{Re} x(n_1, n_2) \Leftrightarrow X_e(\omega_1, \omega_2),$$

$$j \operatorname{Im} x(n_1, n_2) \Leftrightarrow X_o(\omega_1, \omega_2).$$

Symmetry Properties of Real-valued Signals

There are special properties of great interest for *real valued* signals; that is, $x(n_1, n_2) = \operatorname{Re} x(n_1, n_2)$, which implies that $X_o(\omega_1, \omega_2) = 0$. Thus when the signal x is real valued, we must have

$$X(\omega_1, \omega_2) = X^*(-\omega_1, -\omega_2). \quad (1.2-6)$$

Directly from this equation, we get the following transform pairs, or symmetry properties, by taking the real, imaginary, magnitude, and phase of both sides of (1.2–6) and setting the results equal:

$$\operatorname{Re} X(\omega_1, \omega_2) = \operatorname{Re} X(-\omega_1, -\omega_2), \text{ read “real part is even”}$$

$$\operatorname{Im} X(\omega_1, \omega_2) = -\operatorname{Im} X(-\omega_1, -\omega_2), \text{ read “imaginary part is odd”}$$

$$|X(\omega_1, \omega_2)| = |X(-\omega_1, -\omega_2)|, \text{ read “magnitude is even”}$$

$$\arg X(\omega_1, \omega_2) = -\arg X(-\omega_1, -\omega_2), \text{ read “argument (phase) may be taken as odd”}$$

Note that “even” here really means *symmetric through the origin*—i.e., $x(n_1, n_2) = x(-n_1, -n_2)$. Combining results, we see that the 2-D FT of an even real valued signal will be real and even, and this condition is necessary also.

Continuous-Space Fourier Transform

While this course focuses on digital image and video, we need to be aware of the generalization of continuous-time Fourier transforms to two and higher dimensions. Here, we look at the 2-D continuous-parameter Fourier transform, with application to continuous-space images (e.g., film). We will denote the two continuous parameters as t_1 and t_2 , but time is not the target here, although one of the two parameters could be time (e.g., acoustic array processing in geophysics [7]). We could equally have used the notation x_1 and x_2 to denote these free parameters. We have chosen t_1 and t_2 so we can use x for signal.

2-D Fourier Continuous Transform

We start with a continuous-parameter function of two dimensions t_1 and t_2 , properly called a bi-function, and herein denoted $f_c(t_1, t_2)$ defined over $-\infty < t_1 < +\infty, -\infty < t_2 < +\infty$. We write the corresponding 2-D Fourier transform as the double integral

$$F_c(\Omega_1, \Omega_2) \triangleq \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_c(t_1, t_2) \exp -j(\Omega_1 t_1 + \Omega_2 t_2) dt_1 dt_2.$$

We recognize that this FT operator is separable and consists of the 1-D FT repeatedly applied in the t_1 domain for each fixed t_2 , followed by the 1-D FT repeatedly applied in the t_2 domain for each fixed Ω_1 , finally culminating in the value $F_c(\Omega_1, \Omega_2)$. Twice applying the inverse 1-D Fourier theory, once for Ω_1 and once for Ω_2 , we arrive at the inverse continuous-parameter Fourier transform,

$$f_c(t_1, t_2) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F_c(\Omega_1, \Omega_2) \exp +j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2,$$

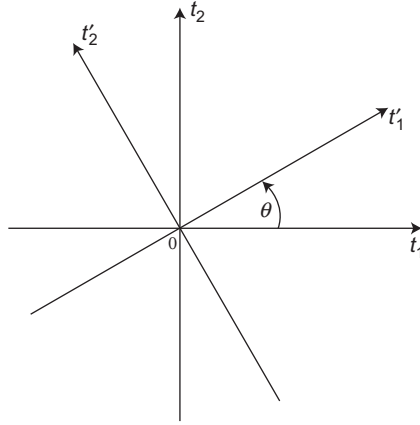
assuming only that the various integrals converge, meaning they are well defined in some sense. One very useful property of the FT operator on such functions is the so-called rotation theorem, which states that the FT of any rotated version of the original function $f_c(t_1, t_2)$ is just the rotated FT—i.e., the corresponding rotation operator applied to $F_c(\Omega_1, \Omega_2)$. We call such an operator *rotationally invariant*. Notationally, we can economically denote such rotations via the matrix equation

$$\mathbf{t}' \triangleq \mathbf{R}\mathbf{t}, \quad (1.2-7)$$

where the vectors are $\mathbf{t} \triangleq (t_1, t_2)^T$, $\mathbf{t}' \triangleq (t'_1, t'_2)^T$, and the rotation matrix \mathbf{R} is given in terms of the rotation angle θ as

$$\mathbf{R} \triangleq \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix},$$

with the sense defined in Figure 1.2-4.

**FIGURE 1.2-4**

Coordinate system t' is rotated from coordinate system t by rotation angle $+\theta$.

Theorem 1.2-2: Rotational Invariance of FT

The continuous-parameter Fourier transform FT is rotationally invariant; that is, if $g_c(t) \triangleq f_c(Rt)$ with R a rotation matrix, then $G_c(\Omega) = F_c(R\Omega)$.

Proof We start at

$$\begin{aligned} g_c(t) &\triangleq f_c(Rt) \\ &= f_c(t_1 \cos \theta + t_2 \sin \theta, -t_1 \sin \theta + t_2 \cos \theta), \end{aligned}$$

which corresponds to a rotation of axes by $+\theta$ degrees.³ Then its Fourier transform is given by

$$\begin{aligned} G_c(\Omega_1, \Omega_2) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g_c(t_1, t_2) \exp -j(\Omega_1 t_1 + \Omega_2 t_2) dt_1 dt_2 \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g_c(t) \exp -j(\Omega^T t) dt \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_c(Rt) \exp -j(\Omega^T t) dt, \end{aligned}$$

³While the axes are rotated by $+\theta$ in going from t to t' , the function $g_c(t) = f_c(Rt) = f_c(t')$, and so the function g_c is a rotated version of f_c with rotation $-\theta$.

where $\mathbf{\Omega} \triangleq (\Omega_1, \Omega_2)^T$. Then we use (1.2-7) to rotate coordinates to get

$$G_c(\Omega_1, \Omega_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_c(\mathbf{t}') \exp -j(\mathbf{\Omega}^T \mathbf{R}^{-1} \mathbf{t}') |\mathbf{R}^{-1}| d\mathbf{t}',$$

where $|\bullet|$ denotes the determinant,

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_c(\mathbf{t}') \exp -j(\mathbf{\Omega}^T \mathbf{R}^T \mathbf{t}') |\mathbf{R}^{-1}| d\mathbf{t}',$$

since the inverse of \mathbf{R} is just its transpose \mathbf{R}^T ,

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_c(\mathbf{t}') \exp -j\{(\mathbf{R}\mathbf{\Omega})^T \mathbf{t}'\} d\mathbf{t}',$$

and the determinant of \mathbf{R} is one,

$$\begin{aligned} &= F_c(\mathbf{R}\mathbf{\Omega}) \\ &= F_c(\Omega_1 \cos \theta + \Omega_2 \sin \theta, -\Omega_1 \sin \theta + \Omega_2 \cos \theta) \\ &= F_c(\mathbf{\Omega}'), \end{aligned}$$

upon setting $\mathbf{\Omega}' \triangleq \mathbf{R}\mathbf{\Omega}$. This completes the proof. ■

Projection-Slice Theorem

In the medical/industrial 3-D imaging technique, computed tomography (CT), two-dimensional slices are constructed from a series of X-ray images taken on an arc. A key element in this approach is the projection-slice theorem presented here. First, we define the Radon transform.

Definition 1.2-4: Radon Transform and Projection

We define the *Radon transform* for angles, $-\pi \leq \theta \leq +\pi$, as

$$p_\theta(t) \triangleq \int_{-\infty}^{+\infty} f_c(t \cos \theta - u \sin \theta, t \sin \theta + u \cos \theta) du,$$

where $f_c(t, u)$ is an integrable and continuous function defined over all of R^2 . ■

Using the rotation matrix \mathbf{R} and writing it as a function of θ , we have $\mathbf{R} = \mathbf{R}(\theta)$ and can express the radon transform as

$$p_\theta(t) = \int_{-\infty}^{+\infty} f_c \left(\mathbf{R}(\theta) \begin{bmatrix} t \\ u \end{bmatrix} \right) du.$$

We can then state the projection-slice theorem.

Theorem 1.2–3: Projection-Slice

Let continuous function $f_c(t, u)$ have Fourier transform $F_c(\Omega_1, \Omega_2)$. Let the Radon transform of f_c be defined and given as $p_\theta(t)$ with 1-D Fourier transform $P_\theta(\Omega)$ for each angle θ . Then we have the equality

$$P_\theta(\Omega) = F_c(\Omega \cos \theta, \Omega \sin \theta). \quad (1.2-8)$$

Proof By the rotational invariance of the continuous FT operator, we only have to show (1.2-8) for $\theta = 0$. So

$$\begin{aligned} P_0(\Omega) &= \text{FT}[p_0(t)] \\ &= \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} f_c(t, u) du \right] e^{-j\Omega t} dt \\ &= F_c(\Omega, 0), \end{aligned}$$

which agrees with (1.2-8) as was to be shown. ■

In words, we can sweep out the full 2-D spatial transform F_c in terms of the 1-D frequency components P_θ of the radon transform at all angles $\theta \in [-\pi, +\pi]$.

CONCLUSIONS

This chapter has introduced 2-D or spatial signals and systems. We looked at how the familiar step and impulse functions of 1-D digital signal processing (DSP) generalize to the 2-D case. We then extended the Fourier transform and studied some of its important properties. We looked at some ideal spatial filters and other important discrete space Fourier transform pairs. We then turned to the continuous-space Fourier transform and treated an application in computer tomography, which is understood using the rotational invariant feature of this transform. The projection-slice theorem is only approximated in the discrete-space Fourier transform and when the sampling rate is high. We discuss sampling in the next chapter. A good alternative reference for these first few chapters is Lim [6].

PROBLEMS

1. Plot the following signals:
 - (a) $s(n_1, n_2) = u(n_1, n_2)u(5 - n_1, n_2)$
 - (b) $s(n_1, n_2) = \delta(n_1 - n_2)$
 - (c) $s(n_1, n_2) = \delta(n_1, n_2 - 1)$
 - (d) $s(n_1, n_2) = \delta(n_1 - 3n_2)u(-n_1, 2 - n_2)$

2. Let a given periodic signal satisfy

$$\begin{aligned} s(n_1, n_2) &= s(n_1 + 5, n_2 + 3) \\ &= s(n_1 - 1, n_2 - 3) \text{ for all } n_1, n_2. \end{aligned}$$

- (a) What is the corresponding periodicity matrix N ?
 (b) Is s periodic also with vectors

$$\mathbf{n}_1 = \begin{bmatrix} 4 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{n}_2 = \begin{bmatrix} 3 \\ -3 \end{bmatrix} ?$$

3. This problem concerns the period of cosine waves in two dimensions, both in continuous space and in discrete space:

- (a) What is the period in continuous space of the cosine wave

$$\cos 2\pi \left(\frac{4t_1 + t_2}{16} \right) ?$$

- (b) What is the period of the cosine wave

$$\cos 2\pi \left(\frac{4n_1 + n_2}{16} \right) ?$$

You may find several answers. Which one is best? Why?

4. Convolve the spatial impulse response

$$h(n_1, n_2) = \begin{cases} \frac{1}{2}, & (n_1, n_2) = (0, 0), (1, 0), (0, 1), \text{ or } (1, 1), \\ 0, & \text{otherwise,} \end{cases}$$

and the input signal

$$x(n_1, n_2) = \begin{cases} 1, & (n_1, n_2) \in [-2, +2] \times [-2, +2], \\ 0, & \text{otherwise.} \end{cases}$$

Call the output signal $y(n_1, n_2)$ and find all of its nonzero values along with their locations.

5. Convolve the spatial impulse response

$$h(n_1, n_2) = \begin{cases} \frac{1}{4}, & (n_1, n_2) = (0, 0), (1, 0), (0, 1), \text{ or } (1, 1), \\ 0, & \text{otherwise,} \end{cases}$$

and the input signal $u_{++}(n_1, n_2)$, the first quadrant unit step function.

6. Which of the following 2-D systems is LSI?

- (a) $y(n_1, n_2) = 3x(n_1, n_2) - x(n_1 - 1, n_2)$
 (b) $y(n_1, n_2) = 3x(n_1, n_2) - y(n_1 - 1, n_2)$ Any additional information needed?
 (c) $y(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{W}(n_1, n_2)} x(k_1, k_2)$ Here, $\mathcal{W}(n_1, n_2) \triangleq \{(n_1, n_2), (n_1 - 1, n_2), (n_1, n_2 - 1), (n_1 - 1, n_2 - 1)\}$.

(d) For the same region $\mathcal{W}(n_1, n_2)$, but now

$$y(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{W}(n_1, n_2)} x(n_1 - k_1, n_2 - k_2)$$

What can you say about the stability of each of these systems?

7. Consider the 2-D signal

$$x(n_1, n_2) = 4 + 2 \cos \left[\frac{2\pi}{8} (n_1 + n_2) \right] + 2 \cos \left[\frac{2\pi}{8} (n_1 - n_2) \right],$$

for all $-\infty < n_1, n_2 < +\infty$. Find the 2-D Fourier transform of x and give a labeled sketch of it.

8. Compute and simplify the Fourier transform of

$$x(n_1, n_2) = 2 \sin(\omega_1^0 n_1 + \omega_2^0 n_2) I_{N \times N}(n_1, n_2),$$

where $I_{N \times N}$ is the indicator function for the $N \times N$ square region $\{0 \leq n_1 \leq N - 1, 0 \leq n_2 \leq N - 1\}$. Simplify your final result as much as possible.

9. In general, the Fourier transform of a product of 2-D sequences corresponds through Fourier transformation to the periodic convolution of their transforms:

$$x(n_1, n_2) y(n_1, n_2) \Leftrightarrow X(\omega_1, \omega_2) \circledast Y(\omega_1, \omega_2).$$

However, if the product is a separable product, then the product of 1-D sequences corresponds through Fourier transformation to the product of their Fourier transforms:

$$x(n_1) y(n_2) \Leftrightarrow X(\omega_1) Y(\omega_2).$$

Reconcile these two facts by writing $x(n_1)$ and $y(n_2)$ as 2-D sequences, taking their 2-D Fourier transforms, and showing that the resulting periodic convolution in the 2-D frequency domain reduces to the product of two 1-D Fourier transforms.

10. Take the inverse Fourier transform of (1.2-3) to check that we obtain the 45° impulse line $\delta(n_1 - n_2)$.

11. For a general complex signal $x(n_1, n_2)$, show that (1.2-5) is correct.

12. Let the signal $x(n_1, n_2) = 1\delta(n_1, n_2) + 0.5\delta(n_1 - 1, n_2) + 0.5\delta(n_1, n_2 - 1)$ have Fourier transform $X(\omega_1, \omega_2)$. What is the inverse Fourier transform of $|X(\omega_1, \omega_2)|^2$?

13. A certain *ideal* lowpass filter with cutoff frequency $\omega_c = \frac{\pi}{2}$ has impulse response

$$h_d(n_1, n_2) = \frac{1}{\sqrt{n_1^2 + n_2^2}} J_1 \left(\frac{\pi}{2} \sqrt{n_1^2 + n_2^2} \right).$$

What is the passband gain of this filter? (Hint: $\lim_{x \rightarrow 0} \frac{J_1(x)}{x} = \frac{1}{2}$)

14. Consider the ideal filter with elliptically shaped frequency response

$$H_e(\omega_1, \omega_2) = \begin{cases} 1 & (\omega_1/\omega_{c1})^2 + (\omega_2/\omega_{c2})^2 \leq 1, \text{ in } [-\pi, +\pi] \times [-\pi, +\pi], \\ 0 & \text{else,} \end{cases}$$

and find the corresponding impulse response $h_e(n_1, n_2)$.

15. We know that the ideal impulse response for a circular support lowpass filter is given in terms of the Bessel function J_1 . Here, we consider the generalization from circle to ellipse.
- (a) What is the impulse response $h_e(n_1, n_2)$ of an elliptical support lowpass filter (see Figure 1.P-1). Assume the major axis cutoff frequency is ω_{c1} and the minor axis cutoff frequency is ω_{c2} , where both cutoff frequencies are positive and less than π . (Hint: Try a transformation of variables on the known circular solution.)

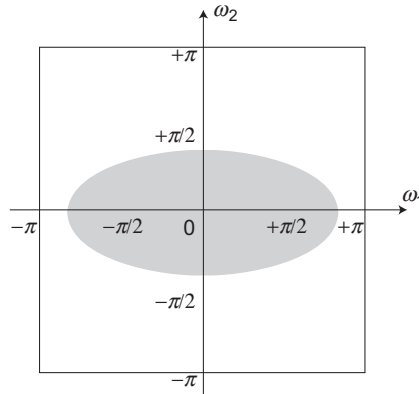
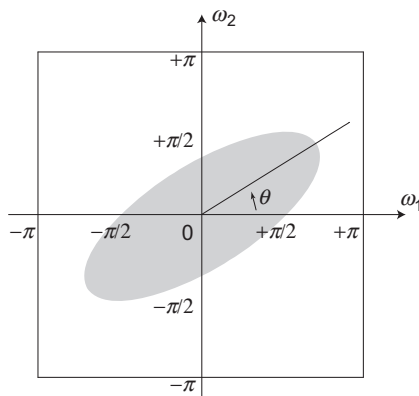


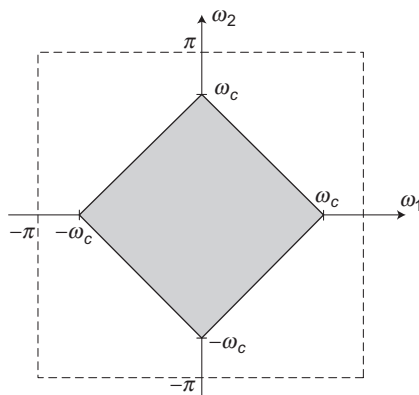
FIGURE 1.P-1

Ideal lowpass filter with elliptical support in the frequency domain, with cutoff frequencies ω_{c1} and ω_{c2} .

- (b) What is the impulse response of the elliptical support lowpass filter after rotation by angle θ (see Figure 1.P-2), where θ is taken as the positive angle between the major axis of the ellipse and the horizontal frequency axis ω_1 . Call the resulting impulse response $h_{e,\theta}(n_1, n_2)$. (Hint: Consider using a rotation matrix as in (1.2-7), but rotation in the frequency domain now.)
16. The ideal frequency-domain diamond lowpass filter with cutoff frequency ω_c satisfying $0 < \omega_c < \pi$ is shown in Figure 1.P-3. Here, the ideal frequency response $H(\omega_1, \omega_2)$ is 1 in the gray region of the figure and is zero elsewhere in $[-\pi, +\pi]^2$. Find the explicit expression in the spatial domain for this filter; i.e., find the ideal diamond lowpass filter impulse response $h(n_1, n_2)$.

**FIGURE 1.P-2**

Ideal filter of Figure 1.P-1 with passband rotated with angle θ as indicated.

**FIGURE 1.P-3**

Ideal diamond lowpass filter with passband cutoff frequency ω_c .

REFERENCES

- [1] A. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
- [2] W. Rudin, *Real and Complex Analysis*, McGraw-Hill, New York, 1966.
- [3] S. Thurnhofer and S. K. Mitra, "A General Framework for Quadratic Volterra Filters for Edge Enhancement," *IEEE Trans. Image Process.*, vol. 5, June, pp. 950–963, 1996.
- [4] F. B. Hildebrand, *Advanced Calculus for Applications*, pp. 254–5, Prentice-Hall, Englewood Cliffs, NJ, 1962.

- [5] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, p. 360 (9.1.18) and p. 484 (11.3.20), Dover, NY, 1965.
- [6] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [7] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Sampling in Two Dimensions

In two- and higher-dimensional signal processing, sampling and the underlying continuous-space play a bigger role than in 1-D signal processing. This is due to the fact that some digital images, and many digital videos, are undersampled. When working with motion in video, half-pixel accuracy is often the minimal accuracy needed. Also there is a variety of sampling possible in space that does not exist in the case of the 1-D time axis. We start out with the so-called rectangular sampling theorem, and then move on to more general but still regular sampling patterns and their corresponding sampling theorems. An example of a regular nonrectangular grid is the hexagonal array of light-receptive cones on the human retina. In spatiotemporal processing, diamond sampling is used in interlaced video formats, where one dimension is the vertical image axis and the second dimension is time. Two-dimensional continuous-space Fourier transform theory is often applied in the study of lens systems in optical devices (e.g., in cameras and projectors), wherein the optical intensity field at a distance of one focal length from the lens is approximated well by the Fourier transform. This study is called Fourier optics, a basic theory used in the design of lenses.

2.1 SAMPLING THEOREM—RECTANGULAR CASE

We assume that the continuous-space function $x_c(t_1, t_2)$ is given. It could correspond to a film image or some other continuous spatial data (e.g., a focused image through a lens system). We can think of the axes t_1 and t_2 as being orthogonal, but that is not necessary. We proceed to produce samples via the rectangular (orthogonal) sampling pattern

$$t_1 = n_1 T_1 \quad \text{and} \quad t_2 = n_2 T_2,$$

thereby producing the discrete-space data

$$x(n_1, n_2) \triangleq x_c(t_1, t_2)|_{t_1=n_1 T_1, t_2=n_2 T_2}.$$

We call this process *rectangular sampling*.

Theorem 2.1–1: Rectangular Sampling

Let $x(n_1, n_2) \triangleq x_c(t_1, t_2)|_{t_1=n_1T_1, t_2=n_2T_2}$, with $T_1, T_2 > 0$, a regular sampling on the continuous-space function x_c on the space (space-time axes) t_1 and t_2 . Then the Fourier transform of the discrete-space sequence $x(n_1, n_2)$ is

$$X(\omega_1, \omega_2) = \frac{1}{T_1 T_2} \sum_{\text{all } k_1, k_2} X_c\left(\frac{\omega_1 - 2\pi k_1}{T_1}, \frac{\omega_2 - 2\pi k_2}{T_2}\right).$$

Proof We start by writing $x(n_1, n_2)$ in terms of the samples of the inverse Fourier transform of $X_c(\Omega_1, \Omega_2)$:

$$x(n_1, n_2) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\Omega_1, \Omega_2) \exp + j(\Omega_1 n_1 T_1 + \Omega_2 n_2 T_2) d\Omega_1 d\Omega_2.$$

Next, we let $\omega_1 \triangleq \Omega_1 T_1$ and $\omega_2 \triangleq \Omega_2 T_2$ in this integral to get

$$\begin{aligned} x(n_1, n_2) &= \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{1}{T_1 T_2} X_c\left(\frac{\omega_1}{T_1}, \frac{\omega_2}{T_2}\right) \exp + j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2 \\ &= \frac{1}{(2\pi)^2} \sum_{\text{all } k_1, k_2} \int_{SQ(k_1, k_2)} \frac{1}{T_1 T_2} X_c\left(\frac{\omega_1}{T_1}, \frac{\omega_2}{T_2}\right) \exp + j(\omega_1 n_1 + \omega_2 n_2) d\omega_1 d\omega_2, \end{aligned} \quad (2.1-1)$$

where $SQ(k_1, k_2)$ is a $2\pi \times 2\pi$ square centered at position $(2\pi k_1, 2\pi k_2)$; that is,

$$SQ(k_1, k_2) \triangleq [-\pi + 2\pi k_1, +\pi + 2\pi k_1] \times [-\pi + 2\pi k_2, +\pi + 2\pi k_2].$$

Then, making the change of variables $\omega'_1 \triangleq \omega_1 - 2\pi k_1$ and $\omega'_2 \triangleq \omega_2 - 2\pi k_2$ separately and *inside* each of the above integrals, we get

$$\begin{aligned} x(n_1, n_2) &= \frac{1}{(2\pi)^2} \sum_{\text{all } k_1, k_2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \frac{1}{T_1 T_2} X_c\left(\frac{\omega'_1 + 2\pi k_1}{T_1}, \frac{\omega'_2 + 2\pi k_2}{T_2}\right) \\ &\quad \times \exp + j(\omega'_1 n_1 + \omega'_2 n_2) d\omega'_1 d\omega'_2 \\ &= \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \left[\sum_{\text{all } k_1, k_2} \frac{1}{T_1 T_2} X_c\left(\frac{\omega'_1 + 2\pi k_1}{T_1}, \frac{\omega'_2 + 2\pi k_2}{T_2}\right) \right] \\ &\quad \times \exp + j(\omega'_1 n_1 + \omega'_2 n_2) d\omega'_1 d\omega'_2 \\ &= \text{IFT} \left[\sum_{\text{all } k_1, k_2} \frac{1}{T_1 T_2} X_c\left(\frac{\omega_1 + 2\pi k_1}{T_1}, \frac{\omega_2 + 2\pi k_2}{T_2}\right) \right] \end{aligned}$$

as was to be shown. ■

Thus we have established the important and basic Fourier sampling relation

$$X(\omega_1, \omega_2) = \frac{1}{T_1 T_2} \sum_{k_1, k_2} X_c \left(\frac{\omega_1 - 2\pi k_1}{T_1}, \frac{\omega_2 - 2\pi k_2}{T_2} \right), \quad (2.1-2)$$

which can also be written in terms of analog frequency as

$$X(T_1 \Omega_1, T_2 \Omega_2) = \frac{1}{T_1 T_2} \sum_{k_1, k_2} X_c \left(\Omega_1 - \frac{2\pi k_1}{T_1}, \Omega_2 - \frac{2\pi k_2}{T_2} \right), \quad (2.1-3)$$

showing more clearly where the aliased components in $X(\omega_1, \omega_2)$ come from in the analog frequency domain. The aliased components are each centered on analog frequency locations $\left(\frac{2\pi k_1}{T_1}, \frac{2\pi k_2}{T_2} \right)$ for all integer grid locations (k_1, k_2) . Of course, for X_c lowpass, and for the case where the sampling density is not very low, we would expect that the main contributions to aliasing would come from the eight nearest neighbor bands corresponding to $(k_1, k_2) = (\pm 1, 0), (0, \pm 1)$, and $(\pm 1, \pm 1)$ in (2.1-2) or (2.1-3), which are sketched in Figure 2.1-1.

This rectangular sampling produces a 2-D spatial frequency aliasing of the continuous-space Fourier transform. As such, we would expect to avoid most of the aliasing, for a nonideal lowpass signal, by choosing the sampling periods T_1, T_2 small

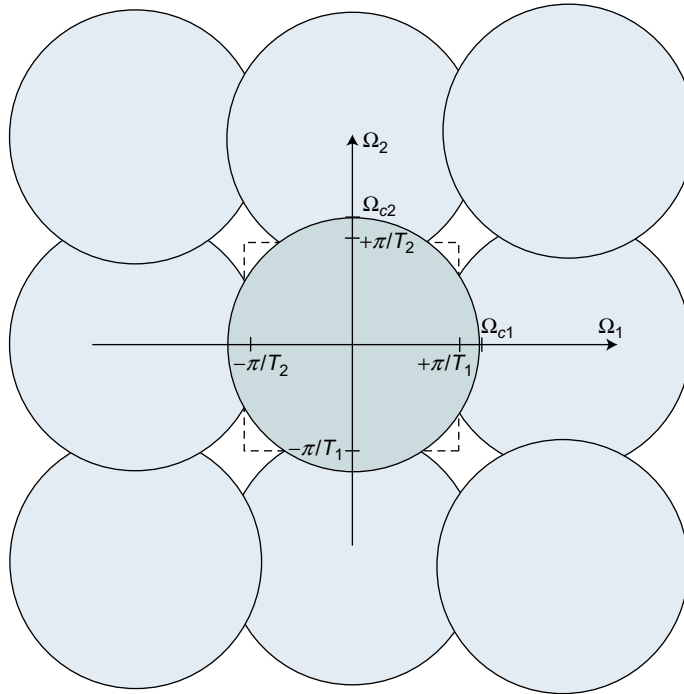


FIGURE 2.1-1

An illustration of the effect of nearest neighbor aliases for the circular symmetric case.

enough. We notice that a variety of aliasing can occur. It can be horizontal, vertical, and/or diagonal aliasing.

We notice that if the input signal is rectangularly bandlimited in the sense that

$$X_c(\Omega_1, \Omega_2) = 0 \quad \text{for} \quad |\Omega_1| \geq \frac{\pi}{T_1} \quad \text{OR} \quad |\Omega_2| \geq \frac{\pi}{T_2}$$

or equivalently that $\text{supp}\{X_c(\Omega_1, \Omega_2)\} = (-\pi/T_1, +\pi/T_1) \times (-\pi/T_2, +\pi/T_2)$, then we have no aliasing in the baseband, and

$$X(\omega_1, \omega_2) = \frac{1}{T_1 T_2} X_c\left(\frac{\omega_1}{T_1}, \frac{\omega_2}{T_2}\right) \quad \text{for} \quad |\omega_1| \leq \pi \quad \text{AND} \quad |\omega_2| \leq \pi,$$

or equivalently in terms of analog frequencies,

$$X(T_1 \Omega_1, T_2 \Omega_2) = \frac{1}{T_1 T_2} X_c(\Omega_1, \Omega_2) \quad \text{for} \quad |T_1 \Omega_1| \leq \pi \quad \text{AND} \quad |T_2 \Omega_2| \leq \pi,$$

so that X_c can be recovered exactly, where it is nonzero, from the Fourier transform of its sampled version, by

$$X_c(\Omega_1, \Omega_2) = T_1 T_2 X(T_1 \Omega_1, T_2 \Omega_2) \quad \text{for} \quad |\Omega_1| \leq \pi/T_1 \quad \text{AND} \quad |\Omega_2| \leq \pi/T_2.$$

More generally, these exact reconstruction results will be true for any signal rectangularly bandlimited to $[-\Omega_{c1}, +\Omega_{c1}] \times [-\Omega_{c2}, +\Omega_{c2}]$ whenever

$$\Omega_{c1} \leq \pi/T_1 \quad \text{AND} \quad \Omega_{c2} \leq \pi/T_2.$$

This is illustrated in Figure 2.1–2 for a circularly bandlimited signal.

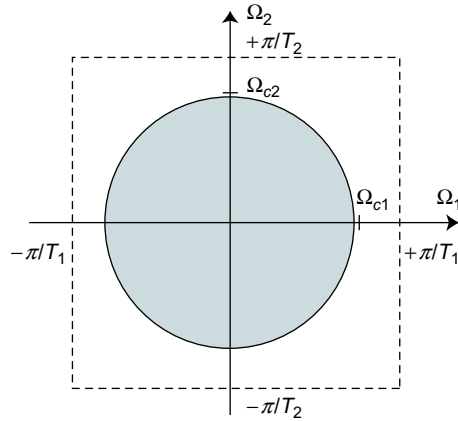


FIGURE 2.1–2

A continuous Fourier transform that will not alias when sampled at all the multiples of (T_1, T_2) .

Reconstruction Formula

At this point we have found the effect of rectangular sampling in the frequency domain. We have seen that no information is lost if the horizontal and vertical sampling rate is high enough for rectangular bandlimited signals. Next, we investigate how to reconstruct the original signal from these samples. To obtain x_c , we start out by writing the continuous-space IFT:

$$\begin{aligned} x_c(t_1, t_2) &= \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\Omega_1, \Omega_2) \times \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2 \\ &= \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} T_1 T_2 X(T_1 \Omega_1, T_2 \Omega_2) I_{\Omega_{c_1}, \Omega_{c_2}}(\Omega_1, \Omega_2) \\ &\quad \times \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2, \end{aligned}$$

where we have made use of the indicator function

$$I_{\Omega_{c_1}, \Omega_{c_2}}(\Omega_1, \Omega_2) \triangleq \begin{cases} 1, & |\Omega_1| < \Omega_{c_1} \quad \text{and} \quad |\Omega_2| < \Omega_{c_2}, \\ 0, & \text{elsewhere.} \end{cases}$$

Continuing,

$$\begin{aligned} x_c(t_1, t_2) &= \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} T_1 T_2 X(T_1 \Omega_1, T_2 \Omega_2) I_{\Omega_{c_1}, \Omega_{c_2}}(\Omega_1, \Omega_2) \\ &\quad \times \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2 \\ &= \frac{T_1 T_2}{(2\pi)^2} \int_{-\Omega_{c_1}}^{+\Omega_{c_1}} \int_{-\Omega_{c_2}}^{+\Omega_{c_2}} X(T_1 \Omega_1, T_2 \Omega_2) \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2. \end{aligned}$$

Next, we substitute X with its FT expression in terms of the samples $x(n_1, n_2)$,

$$X(T_1 \Omega_1, T_2 \Omega_2) = \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \exp - j(T_1 \Omega_1 n_1 + T_2 \Omega_2 n_2),$$

to obtain

$$\begin{aligned} x_c(t_1, t_2) &= \frac{T_1 T_2}{(2\pi)^2} \int_{-\Omega_{c_1}}^{+\Omega_{c_1}} \int_{-\Omega_{c_2}}^{+\Omega_{c_2}} \left[\sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \exp - j(T_1 \Omega_1 n_1 + T_2 \Omega_2 n_2) \right] \\ &\quad \times \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2, \end{aligned}$$

and then interchange the sums and integrals to obtain

$$\begin{aligned}
 x_c(t_1, t_2) &= T_1 T_2 \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \left\{ \frac{1}{(2\pi)^2} \int_{-\Omega_{c_1}}^{+\Omega_{c_1}} \int_{-\Omega_{c_2}}^{+\Omega_{c_2}} \right. \\
 &\quad \left. \times \exp + j[\Omega_1(t_1 - n_1 T_1) + \Omega_2(t_2 - n_2 T_2)] d\Omega_1 d\Omega_2 \right\} \\
 &= T_1 T_2 \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) h(t_1 - n_1 T_1, t_2 - n_2 T_2), \quad (2.1-4)
 \end{aligned}$$

where

$$h(t_1, t_2) \triangleq \frac{\sin \Omega_{c_1} t_1}{\pi t_1} \frac{\sin \Omega_{c_2} t_2}{\pi t_2}, \quad -\infty < t_1, t_2 < +\infty.$$

The interpolation function h is the continuous-space IFT

$$\frac{1}{(2\pi)^2} \int_{-\Omega_{c_1}}^{+\Omega_{c_1}} \int_{-\Omega_{c_2}}^{+\Omega_{c_2}} \exp + j(\Omega_1 t_1 + \Omega_2 t_2) d\Omega_1 d\Omega_2,$$

which is the impulse response of the ideal rectangular lowpass filter

$$I_{\Omega_{c_1}, \Omega_{c_2}}(\Omega_1, \Omega_2) \triangleq \begin{cases} 1, & |\Omega_1| \leq \Omega_{c_1} \quad \text{and} \quad |\Omega_2| \leq \Omega_{c_2}, \\ 0, & \text{elsewhere.} \end{cases}$$

Equation 2.1-4 is known as the *reconstruction formula* for the rectangular sampling theorem and is valid whenever the sampling rate satisfies

$$T_1^{-1} \geq \frac{\Omega_{c_1}}{\pi} \quad \text{and} \quad T_2^{-1} \geq \frac{\Omega_{c_2}}{\pi}.$$

We note that the reconstruction consists of an infinite weighted sum of delayed ideal lowpass filter impulse responses multiplied by a gain term of $T_1 T_2$ centered at each sample location and weighted by the sample value $x(n_1, n_2)$. As in the 1-D case [1], we can write this in terms of filtering as follows. First, we define the continuous-space impulse train, sometimes called the *modulated impulse train*,

$$x_\delta(t_1, t_2) \triangleq \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \delta(t_1 - n_1 T_1, t_2 - n_2 T_2),$$

and then we put this impulse train function through the filter with impulse response $T_1 T_2 h(t_1, t_2)$.

When the sample rate is minimal, we have the *critical sampling* case, and the interpolation function

$$T_1 T_2 h(t_1, t_2) = T_1 T_2 \frac{\sin \Omega_{c_1} t_1}{\pi t_1} \frac{\sin \Omega_{c_2} t_2}{\pi t_2}$$

becomes

$$\begin{aligned} &= T_1 T_2 \frac{\sin \frac{\pi}{T_1} t_1}{\pi t_1} \frac{\sin \frac{\pi}{T_2} t_2}{\pi t_2} \\ &= \frac{\sin \frac{\pi}{T_1} t_1}{\frac{\pi}{T_1} t_1} \frac{\sin \frac{\pi}{T_2} t_2}{\frac{\pi}{T_2} t_2} \\ &= \frac{\sin \Omega_{c_1} t_1}{\Omega_{c_1} t_1} \frac{\sin \Omega_{c_2} t_2}{\Omega_{c_2} t_2}. \end{aligned}$$

For this critical sampling case, the reconstruction formula becomes

$$x_c(t_1, t_2) = \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \frac{\sin \frac{\pi}{T_1} (t_1 - n_1 T_1)}{\frac{\pi}{T_1} (t_1 - n_1 T_1)} \frac{\sin \frac{\pi}{T_2} (t_2 - n_2 T_2)}{\frac{\pi}{T_2} (t_2 - n_2 T_2)} \quad (2.1-5)$$

$$= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) \frac{\sin \Omega_{c_1} (t_1 - n_1 T_1)}{\Omega_{c_1} (t_1 - n_1 T_1)} \frac{\sin \Omega_{c_2} (t_2 - n_2 T_2)}{\Omega_{c_2} (t_2 - n_2 T_2)}, \quad (2.1-6)$$

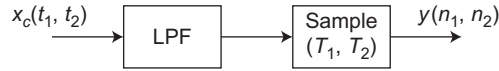
since $\pi/T_i = \Omega_{c_i}$ in this case.

For this critical sampling case, the interpolating functions

$$\frac{\sin \Omega_{c_1} (t_1 - n_1 T_1)}{\Omega_{c_1} (t_1 - n_1 T_1)} \frac{\sin \Omega_{c_2} (t_2 - n_2 T_2)}{\Omega_{c_2} (t_2 - n_2 T_2)}$$

have the property that each one is equal to 1 at its sample location $(n_1 T_1, n_2 T_2)$ and equal to 0 at all other sample locations. Thus at a given sample location, only one of the terms in the double infinite sum is nonzero.

If this critical sampling rate is not achieved, then aliasing can occur as shown in Figure 2.1–1. Note from this figure that if the continuous-space Fourier transform has the indicated circular support, then the main alias contributions will come from the horizontally and vertically nearest neighbor aliased terms; in (2.1–3), the terms correspond to $(k_1, k_2) = (\pm 1, 0)$ and $(0, \pm 1)$, with the diagonal terms at $(\pm 1, \pm 1)$ being the $\sqrt{2}$ further away.

**FIGURE 2.1–3**

A system diagram for *ideal rectangular sampling* to avoid all aliasing error.

Note that even if the analog signal is sampled at high enough rates to avoid any aliasing, there is a type of alias error that can occur on reconstruction due to an inadequate or nonideal reconstruction filter. Sometimes in the 1-D case, this distortion, which is not due to undersampling, is called *imaging distortion* to distinguish it from true aliasing error [2]. This term may not be the best to use in an image and video processing text, so if we refer to this spectral imaging error in the sequel, we will put “image” in quotation marks.

Ideal Rectangular Sampling

In the case where the continuous-space Fourier transform is not bandlimited, one simple expedient is to provide an ideal continuous-space lowpass filter prior to the spatial sampler (Figure 2.1–3). This sampler would be *ideal* in the sense that the maximum possible bandwidth of the signal is preserved alias free.

The lowpass filter (LPF) would pass the maximum band that can be represented with rectangular sampling with period $T_1 \times T_2$, which is the analog frequency band $[-\pi/T_1, +\pi/T_1] \times [-\pi/T_2, +\pi/T_2]$ with passband gain = 1.

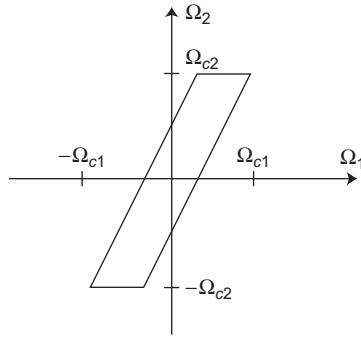
Example 2.1–1: Nonisotropic Signal Spectra

Consider a nonisotropic signal with continuous-space Fourier transform support, as shown in Figure 2.1–4, that could arise from rotation of a texture that is relatively lowpass in one direction and broadband in the perpendicular direction. If we apply the rectangular sampling theorem to this signal, we get the minimal or Nyquist sampling rates

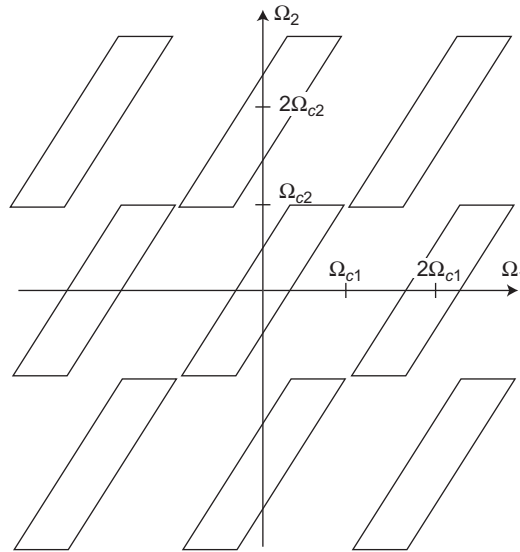
$$T_1^{-1} = \Omega_{c1}/\pi \quad \text{and} \quad T_2^{-1} = \Omega_{c2}/\pi,$$

resulting in the discrete-space Fourier transform support shown in Figure 2.1–5.

Clearly, there is a lot of wasted spectrum here. If we lower the sampling rates judiciously, we can move to a more efficiently sampled discrete-space Fourier transform with support, as shown in Figure 2.1–6. This figure shows aliased replicas that do not overlap, but yet cannot be reconstructed properly with the ideal rectangular reconstruction formula (2.1–5). If we reconstruct with an appropriate ideal diagonal support filter, though, we see that it is still possible to reconstruct this analog signal exactly from this lower

**FIGURE 2.1-4**

A continuous-space Fourier transform with “diagonal” support.

**FIGURE 2.1-5**

The effect of rectangular sampling at rectangular Nyquist rate.

sample-rate data. Effectively, we are changing the basic cell in the analog frequency domain from $[-\pi/T_1, +\pi/T_1] \times [-\pi/T_2, +\pi/T_2]$ to the diagonal basic cell shown in Figure 2.1-7. The dashed-line aliased repeats of the diagonal basic cell show the spectral aliasing resulting from the rectangular sampling, illustrating the resulting periodicity in the sampled Fourier transform.

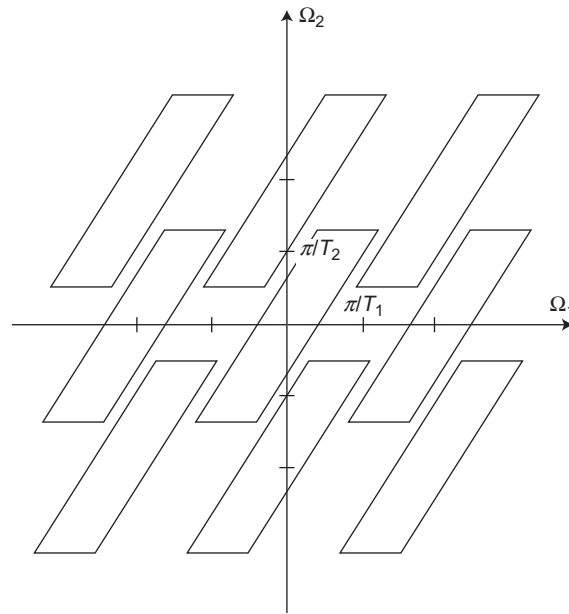


FIGURE 2.1-6

The effect after lowering the vertical sampling rate below the rectangular Nyquist rate.

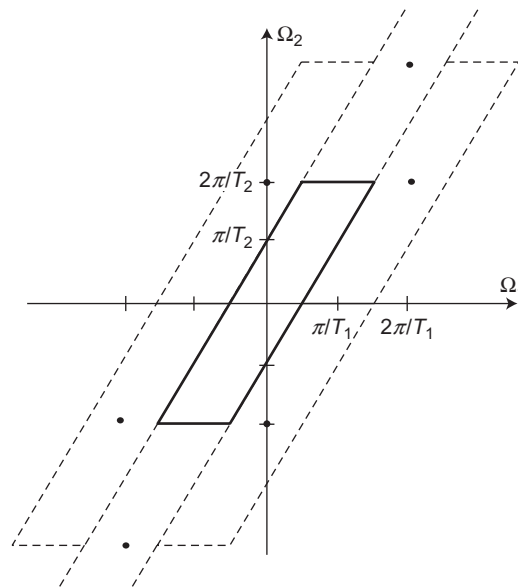


FIGURE 2.1-7

A basic cell, indicated by heavy lines, for diagonal analog Fourier transform support.

From this last example, we see that, more so than in one dimension, there is a wider variety of the support or shape of the incoming analog Fourier-domain data, and it is this analog frequency domain support that, together with the sampling rates, determines whether the resulting discrete-space data are aliased or not. In [Example 2.1–1](#), a rectangular Fourier support would lead to aliased discrete-space data for such low spatial sampling rates, but for the indicated diagonal analog Fourier support shown in [Figure 2.1–6](#), we see that aliasing will not occur if we use this new basic cell. The next example shows one way such diagonal analog frequency domain support arises naturally.

Example 2.1–2: Propagating Plane Waves

Consider the geophysical spatiotemporal data given as

$$s_c(t, x) = g(t - x/v), \quad (2.1-7)$$

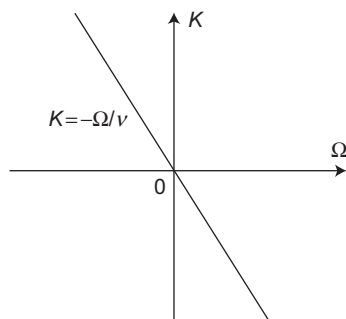
where v is a given velocity, $v \neq 0$. We can interpret this as a plane wave propagating in the $+x$ direction when v is positive, with wave crests given by the equation $t - x/v = \text{constant}$. Here, g is a given function indicating the wave shape. At position $x = 0$, the signal value is just $g(t) = s_c(t, 0)$. At a general position x , we see this same function delayed by the *propagation time* x/v . Taking the continuous parameter time-space Fourier transform, we have

$$\text{FT}[s_c] = \int \int s_c(t, x) \exp -j(\Omega t + Kx) dt dx,$$

where Ω as usual denotes continuous-time radian frequency, and the continuous variable K denotes continuous-space radian frequency referred to as the *wavenumber*. Plugging in the plane-wave [equation \(2.1–7\)](#), we obtain

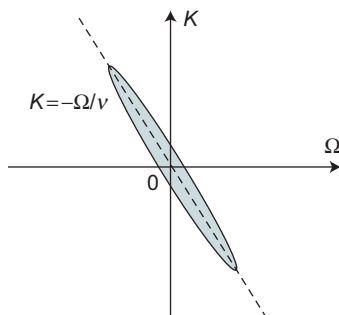
$$\begin{aligned} S_c(\Omega, K) &= \int \int g(t - x/v) \exp -j(\Omega t + Kx) dt dx \\ &= \int \left[\int g(t - x/v) \exp -j\Omega t dt \right] \exp -jKx dx \\ &= \int G(\Omega) \exp (-j\Omega x/v) \exp -jKx dx \\ &= G(\Omega) \int \exp -j(\Omega x/v + Kx) dx \\ &= 2\pi G(\Omega) \delta(K + \Omega/v), \end{aligned}$$

a delta function in the frequency domain, concentrated along the line $K + \Omega/v = 0$, plotted in [Figure 2.1–8](#).

**FIGURE 2.1-8**

Fourier transform illustration of ideal plane wave at velocity $v > 0$.

If we relax the assumption of an exact plane wave in [Example 2.1-2](#), we get some spread out from this ideal impulse line, and hence find a diagonal Fourier transform support as illustrated in [Figure 2.1-9](#). More on the application of multidimensional geophysical processing is contained in [3].

**FIGURE 2.1-9**

Fourier transform illustration of approximate plane wave at velocity v .

Example 2.1-3: Alias Error in Images

In image processing, aliasing energy looks like ringing that is perpendicular to high frequency or sharp edges. This can be seen in [Figure 2.1-10](#), where there was excessive high-frequency information around the palm fronds. We can see the aliasing in the zoomed-in image shown in [Figure 2.1-11](#), where we see ringing parallel to the fronds, caused by some prior filtering, and aliased energy approximately perpendicular to the edges, appearing as a ringing approximately perpendicular to the fronds. A small amount of aliasing is usually not very annoying in images.

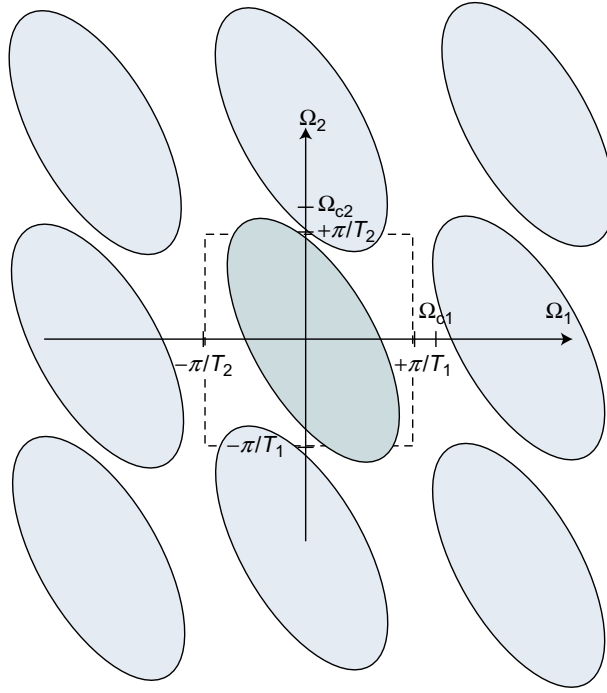
**FIGURE 2.1–10**

2000 × 1000-pixel image with aliasing.

**FIGURE 2.1–11**

Zoomed-in section of aliased image.

To appreciate how the aliasing (or “imaging” error) energy can appear nearly perpendicular to a local diagonal component, refer to [Figure 2.1–12](#), where we see two alias components coming from above and below in opposite quadrants to those of the main signal energy, giving the alias error signal a distinct high-frequency directional component.

**FIGURE 2.1-12**

An illustration of how an alias (or “imaging”) error can arise in a more directional case.

2.2 SAMPLING THEOREM—GENERAL REGULAR CASE

Now consider more general, but still regular, nonorthogonal sampling on the regular grid of locations or *lattice*,

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = [\mathbf{v}_1 \ \mathbf{v}_2] \begin{bmatrix} n_1 \\ n_2 \end{bmatrix},$$

for sampling vectors \mathbf{v}_1 and \mathbf{v}_2 , or what is the same $\mathbf{t} = n_1 \mathbf{v}_1 + n_2 \mathbf{v}_2$ for all integers n_1 and n_2 . Thus we have the sampled data

$$x(\mathbf{n}) \triangleq x_c(\mathbf{V}\mathbf{n}), \quad (2.2-1)$$

with *sampling matrix* $\mathbf{V} \triangleq [\mathbf{v}_1 \ \mathbf{v}_2]$. The sampling matrix is assumed always invertible, since otherwise, the sampling locations would not cover the plane (i.e., would not be a lattice). The rectangular or Cartesian sampling we encountered in the previous section is the special case $\mathbf{v}_1 = (T_1, 0)^T$ and $\mathbf{v}_2 = (0, T_2)^T$.

Example 2.2–1: Hexagonal Sampling Lattice

A sampling matrix of particular importance,

$$V = \begin{bmatrix} 1 & 1 \\ 1/\sqrt{3} & -1/\sqrt{3} \end{bmatrix},$$

results in a hexagonal sampling pattern. The resulting hexagonal sampling pattern is sketched in Figure 2.2–1, where we note that the axes n_1 and n_2 are no longer orthogonal to one another. The hexagonal sampling grid shape comes from the fact that the two sampling vectors have angle $\pm 30^\circ$ with the horizontal axis, and that they are of equal length. It is easily seen by example, though, that these sample vectors are not unique to produce this grid. One could as well use $\mathbf{v}_1 = (0, 1/\sqrt{3})^T$ together with $\mathbf{v}_2 = (0, 2/\sqrt{3})^T$ to generate this same lattice. We will see later that the hexagonal sampling grid can be more efficient than the Cartesian grid in many common image-processing situations.

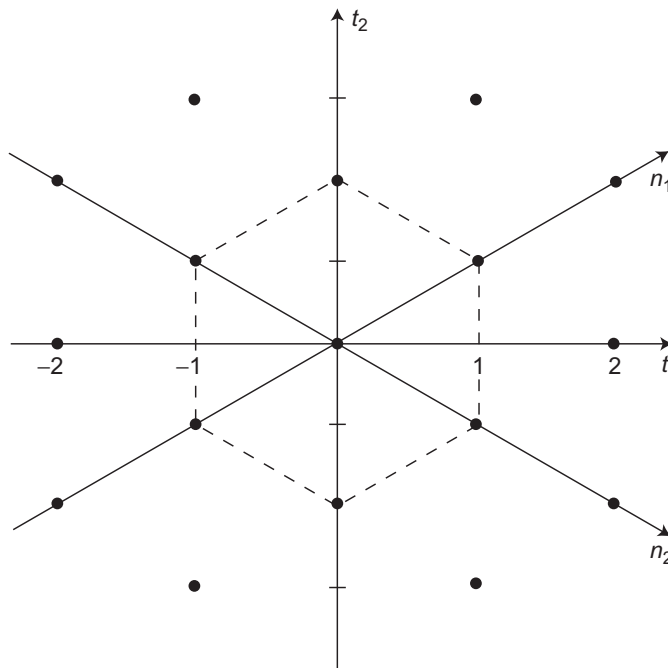


FIGURE 2.2–1

Hexagonal sampling grid in space.

To develop a theory for the general sampling case of (2.2-1), we start by writing the continuous-space inverse Fourier transform,

$$x_c(\mathbf{t}) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\boldsymbol{\Omega}) \exp + j(\boldsymbol{\Omega}^T \mathbf{t}) d\boldsymbol{\Omega},$$

with analog frequency $\boldsymbol{\Omega} \triangleq (\Omega_1, \Omega_2)^T$. Then, by definition of the sample locations, we have the discrete data as

$$x(\mathbf{n}) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\boldsymbol{\Omega}) \exp + j(\boldsymbol{\Omega}^T \mathbf{V}\mathbf{n}) d\boldsymbol{\Omega},$$

which upon writing $\boldsymbol{\omega} \triangleq \mathbf{V}^T \boldsymbol{\Omega}$ becomes

$$x(\mathbf{n}) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\mathbf{V}^{-T} \boldsymbol{\omega}) \exp + j(\boldsymbol{\omega}^T \mathbf{n}) \frac{d\boldsymbol{\omega}}{|\det \mathbf{V}|}. \quad (2.2-2)$$

Here, \mathbf{V}^{-T} denotes the inverse of the transpose sampling matrix \mathbf{V}^T , where the notational simplification is permitted because the order of transpose and inverse commutes for invertible matrices. Next, we break up the integration region in this equation into squares of support $[-\pi, +\pi]^2 \triangleq [-\pi, +\pi] \times [-\pi, +\pi]$ and write

$$x(\mathbf{n}) = \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \frac{1}{|\det \mathbf{V}|} \left\{ \sum_{\text{all } k} X_c[\mathbf{V}^{-T}(\boldsymbol{\omega} - 2\pi \mathbf{k})] \right\} \exp + j(\boldsymbol{\omega}^T \mathbf{n}) d\boldsymbol{\omega},$$

just as in (2.1-1) of the last section, and valid for the same reason. We can now invoke the uniqueness of the inverse Fourier transform for 2-D discrete space to conclude that the discrete-space Fourier transform $X(\boldsymbol{\omega})$ must be given as

$$X(\boldsymbol{\omega}) = \frac{1}{|\det \mathbf{V}|} \left\{ \sum_{\text{all } k} X_c[\mathbf{V}^{-T}(\boldsymbol{\omega} - 2\pi \mathbf{k})] \right\},$$

where the discrete-space Fourier transform $X(\boldsymbol{\omega})$ is $\sum_{\mathbf{n}} x(\mathbf{n}) \exp(-j\boldsymbol{\omega}^T \mathbf{n})$, just as usual.

We now introduce the *periodicity matrix* $\mathbf{U} \triangleq 2\pi \mathbf{V}^{-T}$ and note the fundamental equation

$$\mathbf{U}^T \mathbf{V} = 2\pi \mathbf{I}, \quad (2.2-3)$$

where \mathbf{I} is the identity matrix. This equation relates any regular sampling matrix and its corresponding periodicity matrix in the analog frequency domain.

In terms of periodicity matrix \mathbf{U} , we can write

$$X(\boldsymbol{\omega}) = \frac{1}{|\det \mathbf{V}|} \left\{ \sum_{\text{all } k} X_c \left[\frac{1}{2\pi} \mathbf{U}(\boldsymbol{\omega} - 2\pi \mathbf{k}) \right] \right\}, \quad (2.2-4)$$

which can be written also in terms of the analog frequency variable $\boldsymbol{\Omega}$

$$X(\mathbf{V}^T \boldsymbol{\Omega}) = \frac{1}{|\det \mathbf{V}|} \left\{ \sum_{\text{all } \mathbf{k}} X_c(\boldsymbol{\Omega} - \mathbf{U}\mathbf{k}) \right\},$$

showing that the alias location points of the discrete-space Fourier transform have the periodicity matrix \mathbf{U} when written in the analog frequency variable $\boldsymbol{\Omega}$.

For the rectangular sampling case with sampling matrix

$$\mathbf{V} = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix},$$

the periodicity matrix is

$$\mathbf{U} = \begin{bmatrix} 2\pi/T_1 & 0 \\ 0 & 2\pi/T_2 \end{bmatrix},$$

showing consistency with the results of the previous section, i.e., (2.2–4) simplifies to (2.1–2).

Example 2.2–2: General Hexagonal Case

In the case where the sampling matrix is hexagonal,

$$\mathbf{V} = \begin{bmatrix} T & T \\ T/\sqrt{3} & -T/\sqrt{3} \end{bmatrix},$$

with T being an arbitrary scale coefficient for the sampling vectors. The corresponding periodicity matrix is easily seen to be

$$\mathbf{U} = \begin{bmatrix} \pi/T & \pi/T \\ \pi\sqrt{3}/T & -\pi\sqrt{3}/T \end{bmatrix} = [\mathbf{u}_1, \mathbf{u}_2]$$

and $|\det \mathbf{V}| = 2T^2/\sqrt{3}$. The resulting repetition or alias anchor points in analog frequency space are shown in Figure 2.2–2. Note that the hexagon appears rotated (by 30°) from the one hexagonal sampling grid in the spatial dimension in Figure 2.2–1. At each alias anchor point, the analog Fourier transform would be seated. Aliasing will not occur if the support of the analog Fourier transform is circular and less than $\pi\sqrt{3}/T$ in radius. ■

We next turn to the reconstruction formula for the case of hexagonal sampling.

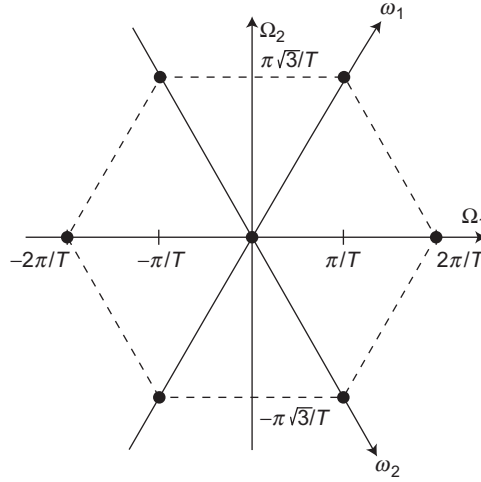


FIGURE 2.2-2

Hexagonal alias repeat grid in analog frequency domain.

Hexagonal Reconstruction Formula

First, we have to scale down the hexagonal grid with scaling parameter T until there is no aliasing (assuming finite analog frequency domain support). Then we have

$$\begin{aligned} X(\mathbf{V}^T \boldsymbol{\Omega}) &= \frac{1}{|\det \mathbf{V}|} X_c(\boldsymbol{\Omega}) \\ &= \frac{\sqrt{3}}{2T^2} X_c(\boldsymbol{\Omega}), \end{aligned}$$

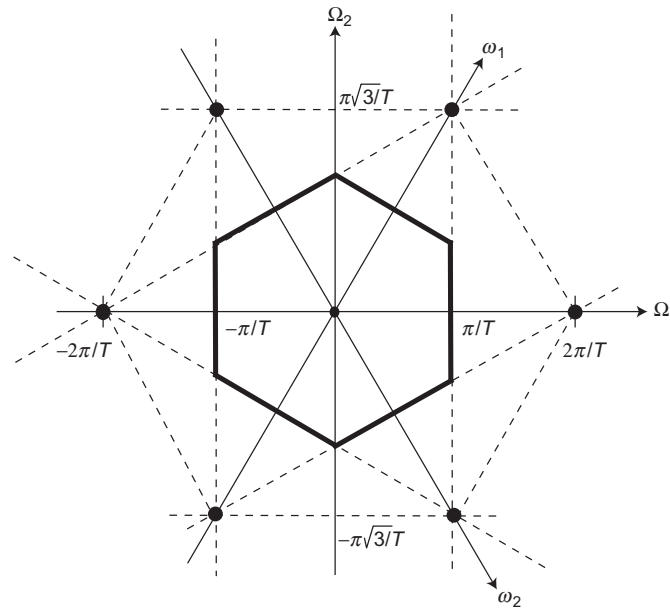
and so

$$x_c(t) = \frac{2T^2}{\sqrt{3}} \left(\frac{1}{2\pi} \right)^2 \sum_n x(n) \int_{\mathcal{B}} \exp[j\boldsymbol{\Omega}^T(t - \mathbf{V}n)] d\boldsymbol{\Omega},$$

where \mathcal{B} is the hexagonal frequency domain basic cell shown in Figure 2.2-3. This basic cell is determined by bounding planes placed to bisect lines joining spectral alias anchor points and the origin. The whole $\boldsymbol{\Omega}$ plane can be covered without gaps or overlaps by repeating this basic cell centered at each anchor point (*tessellated*). It is worthwhile noting that an analog Fourier transform with circular support would fit nicely in such a hexagonal basic cell.

Upon defining the spatial impulse response

$$h(\mathbf{t}) \triangleq \frac{2T^2}{\sqrt{3}} \left(\frac{1}{2\pi} \right)^2 \int_{\mathcal{B}} \exp(j\boldsymbol{\Omega}^T \mathbf{t}) d\boldsymbol{\Omega},$$

**FIGURE 2.2-3**

Hexagonal basic cell in analog frequency domain (heavy line).

we can write the reconstruction formula for hexagonal sampling as

$$x_c(t) = \sum_n x(n)h(t - Vn).$$

Of course, we could subsequently evaluate t on a different sampling lattice and thereby have a way to convert between sampling lattices in two dimensions.

Example 2.2-3: Sampling Efficiency for Spherical Baseband

We can generalize the hexagonal lattice to three dimensions, where it is called a 3-D rhombic dodecahedron, and continue to four and higher dimensions. We would notice then that a spherical baseband fits snugly into a generalized hexagon, much better than into a generalized square or cube. Dudgeon and Mersereau [3] have obtained the following results showing the sampling efficiency of these generalized hexagonal lattices with respect to the Cartesian lattice:

Dimension M	1	2	3	4
Efficiency	1.000	0.866	0.705	0.505

We see a 30% advantage for the 3-D case (e.g., spatiotemporal or 3-D space) and a 50% advantage in 4-D (e.g., the three spatial dimensions + time, or true 3-D video).

In spite of its increased sampling efficiency, the 3-D generalization of the hexagonal lattice has not found much use in applications that sample 3-D data. However, a variation of the diamond sampling lattice has found application in the interlaced sensors, commonly found in video cameras, both standard definition (SD) and high definition (HD).

Example 2.2–4: Diamond-shaped Sampling Lattice

A special sampling lattice is given by sampling matrix

$$\mathbf{V} = \begin{bmatrix} T & T \\ T & -T \end{bmatrix} \text{ and periodicity matrix } \mathbf{U} = \begin{bmatrix} \pi/T & \pi/T \\ \pi/T & -\pi/T \end{bmatrix},$$

resulting in the so-called diamond-shaped lattice, illustrated in Figure 2.2–4 for the normalized case $T = 1$.

If we look to the periodicity matrix in analog frequency space, we obtain Figure 2.2–5.

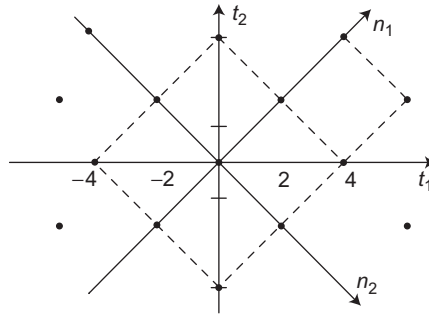


FIGURE 2.2–4

Diamond sampling grid with $T = 1$.

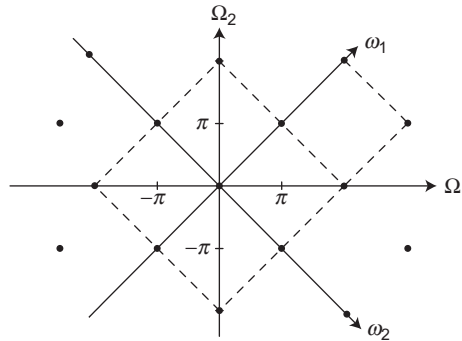


FIGURE 2.2–5

Alias anchor points for diamond sampling with $T = 1$.

If we restrict our 2-D sampling to the vertical and temporal axes of video data, we get the following example of diamond sampling in conventional video.

Example 2.2–5: Interlaced Video

Consider a video signal $s_c(x, y, t)$, sampled on a rectangular or Cartesian grid, to get the so-called *progressive* digital video signal $s(n_1, n_2, n) \triangleq s_c(n_1 \Delta_1, n_2 \Delta_2, n \Delta)$, where $(\Delta_1, \Delta_2)^T$ is the sample vector in space and Δ is the sampling interval in time. But in SD and HD video, it is also common for it to be sampled on an *interlaced* lattice consisting of two *fields* making up each *frame*. One field just contains the even lines, while the second field gets the odd scan lines. In equations, we have

$$s(n_1, 2n_2, 2n) \triangleq s_c(n_1 \Delta_1, 2n_2 \Delta_2, 2n \Delta) \quad \text{and}$$

$$s(n_1, 2n_2 + 1, 2n + 1) \triangleq s_c(n_1 \Delta_1, (2n_2 + 1) \Delta_2, (2n + 1) \Delta)$$

whose sampling period is illustrated in Figure 2.2–6, where we only show the vertical and time axes. We can fix the horizontal variable at, say, $n_1 = n_1^0$ and then regard the data as 2-D (vertical-time) data. In the two mentioned cases, this becomes *progressive*:

$$g_c(y, t) \triangleq s_c(x^0, y, t) \quad \text{and}$$

$$g(n_2, n) = g_c(n_2 \Delta_2, n \Delta),$$

or *interlaced*:

$$g(2n_2, 2n) \triangleq g_c(2n_2 \Delta_2, 2n \Delta) \quad \text{and}$$

$$g(2n_2 + 1, 2n + 1) \triangleq g_c((2n_2 + 1) \Delta_2, (2n + 1) \Delta).$$

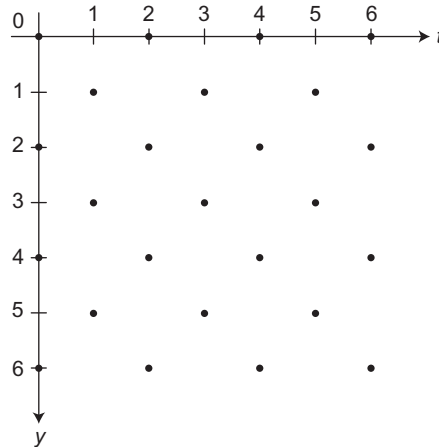


FIGURE 2.2–6

An illustration of interlaced sampling in a 2-D vertical-temporal domain.

The sampling matrix for this 2-D interlaced data is easily seen to be

$$\mathbf{V} = \begin{bmatrix} \Delta_2 & \Delta_2 \\ \Delta & -\Delta \end{bmatrix},$$

corresponding to diamond sampling in the vertical-time domain. Applying the diamond-shaped sample lattice theory, we get a diamond or rotated square baseband to keep alias free. So, if sampled at a high enough spatiotemporal rate, we can reconstruct the continuous space-time signal from the diamond sampled or interlaced digital video data. We would do this, conceptually at least, via 2-D processing in the vertical-temporal domain, for each horizontal value n_1 . ■

In performing the processing in [Example 2.2–5](#), we consider the 3-D spatiotemporal data as a set of 2-D $y \times t$ data planes at each horizontal location $x = x^0$. Thus the 2-D processing would be done on each such $y \times t$ plane separately, and then all the results would be combined. In practice, since there is no interaction between the $y \times t$ planes, this processing is performed interleaved in time, so that the data are processed in time locality (i.e., those frames near frame n are processed together).

Interlace has gotten a bad name in video, despite its increased spectral efficiency over progressive sampling. This is because of three reasons: First, the proper space-time filtering to reconstruct the interlaced video is hardly ever done, and in its place is inserted a lowpass vertical filter, sacrificing typically about 30% of the vertical resolution. Second, almost all flat-panel display devices are progressive; thus interlaced images have to be transformed to progressive by the display hardware, often with less than optimal results, in addition to the vertical resolution loss previously mentioned. Lastly, today arguably the best video sources, both film and digital, are progressive, not interlaced.

2.3 CHANGE OF SAMPLE RATE

Returning to the more common case of rectangular sampling, it is often necessary in image and video signal processing to change the sample rate, either up or down. This is commonly the case in the resizing of images and videos for display on a digital image or video monitor.

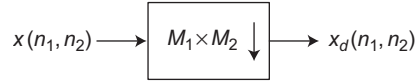
Downsampling by Integers $M_1 \times M_2$

We define decimation or downsampling by integers M_1 and M_2 as follows:

$$x_d(n_1, n_2) \triangleq x(M_1 n_1, M_2 n_2),$$

denoted via the system element shown in [Figure 2.3–1](#). The correspondence to rectangular downsampling in the frequency domain is

$$X_d(\omega_1, \omega_2) = \frac{1}{M_1 M_2} \sum_{i_1=0}^{M_1-1} \sum_{i_2=0}^{M_2-1} X\left(\frac{\omega_1 - 2\pi i_1}{M_1}, \frac{\omega_2 - 2\pi i_2}{M_2}\right). \quad (2.3-1)$$

**FIGURE 2.3–1**

Downsample system element.

One way of deriving this equation is to first imagine an underlying continuous-space function $x_c(t_1, t_2)$, of which the values $x(n_1, n_2)$ are its samples on the grid $T_1 \times T_2$. Then we can regard the downsampled signal x_d as the result of sampling x_c on the less dense sample grid $M_1 T_1 \times M_2 T_2$. From the rectangular sampling theorem, we have

$$X_d(\omega_1, \omega_2) = \frac{1}{M_1 T_1 M_2 T_2} \sum_{k_1, k_2=-\infty}^{+\infty} X_c\left(\frac{\omega_1 - 2\pi k_1}{M_1 T_1}, \frac{\omega_2 - 2\pi k_2}{M_2 T_2}\right). \quad (2.3-2)$$

Comparing this equation with (2.1–2), we can see that it includes the alias components of (2.3–2) when $k_1 = M_1 l_1$ and $k_2 = M_2 l_2$ with l_1 and l_2 integers, plus many others in between. So we can substitute the following sums into (2.3–2):

$$k_1 = i_1 + M_1 l_1, \quad \text{where } i_1 : 0, \dots, M_1 - 1,$$

$$k_2 = i_2 + M_2 l_2, \quad \text{where } i_2 : 0, \dots, M_2 - 1,$$

and then rewrite the equation as

$$\begin{aligned} X_d(\omega_1, \omega_2) &= \frac{1}{M_1 T_1 M_2 T_2} \sum_{l_1, l_2=-\infty}^{+\infty} \sum_{i_1=0}^{M_1-1} \sum_{i_2=0}^{M_2-1} \\ &\quad \times X_c\left(\frac{\omega_1 - 2\pi i_1}{M_1 T_1} - \frac{2\pi l_1}{T_1}, \frac{\omega_2 - 2\pi i_2}{M_2 T_2} - \frac{2\pi l_2}{T_2}\right) \\ &= \frac{1}{M_1 M_2} \sum_{i_1=0}^{M_1-1} \sum_{i_2=0}^{M_2-1} \frac{1}{T_1 T_2} \sum_{l_1, l_2=-\infty}^{+\infty} \\ &\quad \times X_c\left(\frac{\omega_1 - 2\pi i_1}{M_1 T_1} - \frac{2\pi l_1}{T_1}, \frac{\omega_2 - 2\pi i_2}{M_2 T_2} - \frac{2\pi l_2}{T_2}\right) \\ &= \frac{1}{M_1 M_2} \sum_{i_1=0}^{M_1-1} \sum_{i_2=0}^{M_2-1} X\left(\frac{\omega_1 - 2\pi i_1}{M_1}, \frac{\omega_2 - 2\pi i_2}{M_2}\right), \end{aligned} \quad (2.3-3)$$

in terms of the discrete-space Fourier transform X , by evaluation of (2.1–2) at the locations $\left(\frac{\omega_1 - 2\pi i_1}{M_1}, \frac{\omega_2 - 2\pi i_2}{M_2}\right)$.

Example 2.3–1: $M_1 = M_2 = 2$ Case for 2×2 Decimation

In this case, we have $x_d(n_1, n_2) \triangleq x(2n_1, 2n_2)$; thus in the frequency domain with $M_1 = M_2 = 2$, (2.3–3) becomes

$$\begin{aligned} X_d(\omega_1, \omega_2) &= \frac{1}{4} \sum_{i_1=0}^1 \sum_{i_2=0}^1 X\left(\frac{\omega_1 - 2\pi i_1}{2}, \frac{\omega_2 - 2\pi i_2}{2}\right) \\ &= \frac{1}{4} \left[X\left(\frac{\omega_1}{2}, \frac{\omega_2}{2}\right) + X\left(\frac{\omega_1}{2}, \frac{\omega_2}{2} - \pi\right) \right. \\ &\quad \left. + X\left(\frac{\omega_1}{2} - \pi, \frac{\omega_2}{2}\right) + X\left(\frac{\omega_1}{2} - \pi, \frac{\omega_2}{2} - \pi\right) \right]. \end{aligned} \quad (2.3-4)$$

Figure 2.3–2 shows the characteristic periodic support pattern of a lowpass Fourier transform $X(\omega_1, \omega_2)$ plotted out to beyond 2π in each variable. Figure 2.3–3 shows the corresponding Fourier transform $X(\frac{1}{2}\omega_1, \frac{1}{2}\omega_2)$ plotted out to a range beyond 4π in each variable. We see how the periodic support pattern has spread out now, and note that this function has a period of $4\pi \times 4\pi$; therefore, by itself, it cannot be a valid Fourier transform. With reference to this figure, we can see how the other three terms in (2.3–4) can fit in without overlap, each centered at $(0, 2\pi)$, $(2\pi, 0)$, and $(2\pi, 2\pi)$, respectively, to restore the $2\pi \times 2\pi$ periodicity and make X_d a valid Fourier transform. A necessary condition of no-overlap is seen to be $\text{supp}\{X\} \subseteq (-\frac{1}{2}\pi, +\frac{1}{2}\pi) \times (-\frac{1}{2}\pi, +\frac{1}{2}\pi)$. If this condition is met, as shown in Figure 2.3–2, then no information is lost and perfect reconstruction is possible from the subsampled signal.

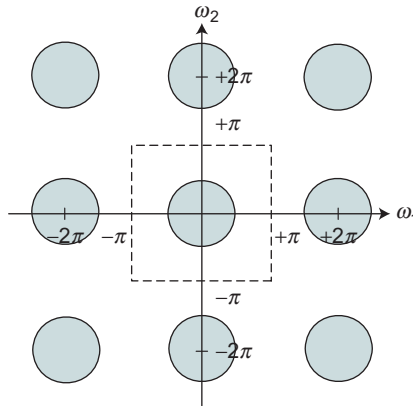
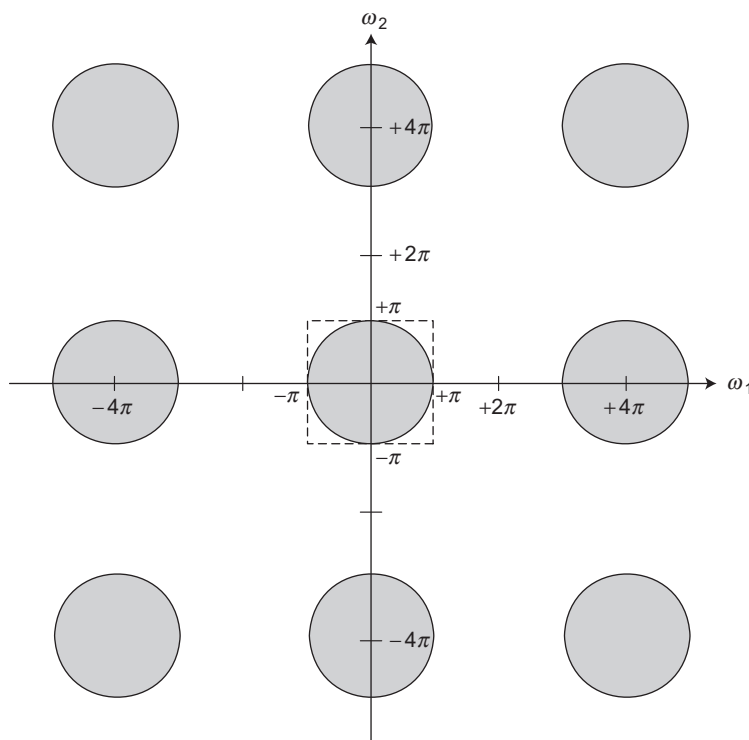
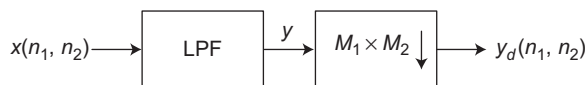
**FIGURE 2.3–2**

Illustration of lowpass X , plotted out beyond 2π in each variable.

**FIGURE 2.3-3**

Fourier transform $X\left(\frac{1}{2}\omega_1, \frac{1}{2}\omega_2\right)$ plotted out to a range beyond 4π in each variable.

**FIGURE 2.3-4**

System diagram for ideal decimation, which avoids aliasing error in the decimated signal.

Ideal Decimation

In order to avoid alias error in the case of a general signal x and to preserve as much of its frequency content as possible, we can choose to insert an ideal rectangular lowpass filter ahead of the downsampler, as shown in Figure 2.3-4. Here, the passband of the filter should have gain 1, and the passband is chosen as $[-\pi/M_1, +\pi/M_1] \times [-\pi/M_2, +\pi/M_2]$ to avoid aliasing in the decimated signal. This process has been called *ideal decimation*.

Example 2.3–2: Ideal 2 × 2 Decimation

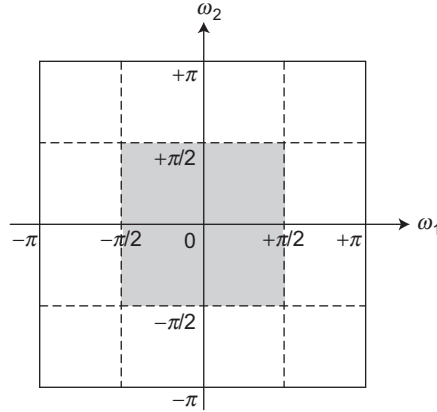
We have a filtered signal here, so upon 2 × 2 decimation, $x_d(n_1, n_2) \triangleq (h * x)(2n_1, 2n_2)$, and in the frequency domain¹

$$\begin{aligned} X_d(\omega_1, \omega_2) &= \frac{1}{4} \sum_{i_1=0}^1 \sum_{i_2=0}^1 (HX) \left(\frac{\omega_1 - 2\pi i_1}{2}, \frac{\omega_2 - 2\pi i_2}{2} \right) \\ &= \frac{1}{4} \left[(HX) \left(\frac{\omega_1}{2}, \frac{\omega_2}{2} \right) + (HX) \left(\frac{\omega_1}{2}, \frac{\omega_2}{2} - \pi \right) + (HX) \left(\frac{\omega_1}{2} - \pi, \frac{\omega_2}{2} \right) \right. \\ &\quad \left. + (HX) \left(\frac{\omega_1}{2} - \pi, \frac{\omega_2}{2} - \pi \right) \right]. \end{aligned} \quad (2.3-5)$$

We can see that the baseband that can be preserved alias free in the decimated signal x_d is $[-\pi/2, +\pi/2] \times [-\pi/2, +\pi/2]$, which is 1/4 of the full bandwidth. This region is shown as the gray area in Figure 2.3–5, a *subband* of the original fullband signal HX . Since this subband is lowpass in each frequency variable, it is called the *LL subband*. If we prefilter with an ideal lowpass filter with gain 1, whose frequency domain support is shown as gray in Figure 2.3–5,

$$H(\omega_1, \omega_2) = \begin{cases} 1, & (\omega_1, \omega_2) \in [-\pi/2, +\pi/2] \times [-\pi/2, +\pi/2], \\ 0, & \text{else,} \end{cases}$$

then we can preserve the LL subband $[-\pi/2, +\pi/2] \times [-\pi/2, +\pi/2]$ from the original signal X .

**FIGURE 2.3–5**

The gray area is the LL subband preserved under 2 × 2 ideal decimation.

¹Here, $(h * x)$ denotes the resulting function of convolution, and (HX) denotes the function resulting from pointwise product.

The higher frequency components or subbands of the original signal X can be obtained as follows. Consider the gray area shown in Figure 2.3–6. We can separate out this part, called the *HL subband*, by prefiltering before the decimation with the ideal bandpass filter, with the passband shown as the gray region in Figure 2.3–6. Similarly, the *LH* and *HH* subbands can be separated out, using filters with supports as shown in Figures 2.3–7 and 2.3–8, respectively. Together, these four subbands contain all the information from the original fullband signal.²

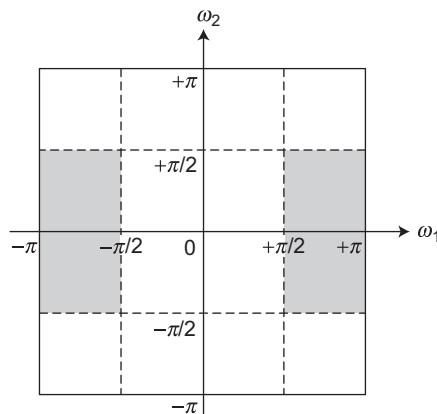


FIGURE 2.3–6

Frequency domain support of the HL subband.

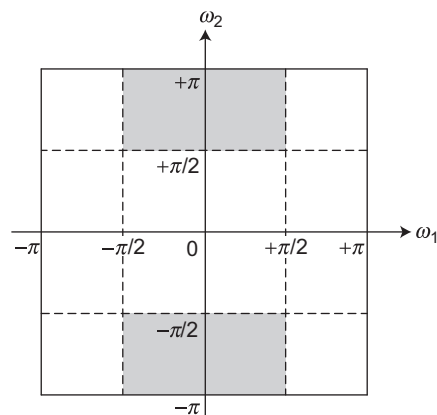
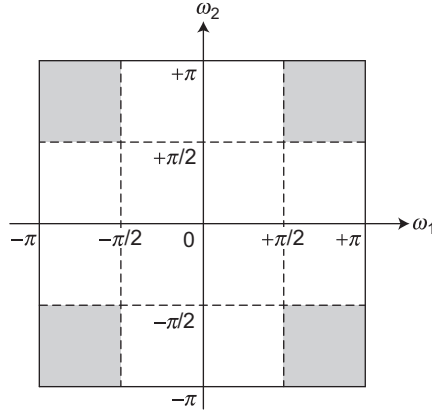


FIGURE 2.3–7

Frequency domain support of the LH subband.

²Care must be taken on the boundaries of these subbands so that information is not duplicated or lost by this ideal filter decomposition.

**FIGURE 2.3-8**

Frequency domain support of the HH subband.

We will see in the sequel that decomposition of a fullband signal into its subbands can offer advantages for both image processing and compression. In fact, the international image compression standard JPEG 2000 is based on subband analysis.

Upsampling by Integers $L_1 \times L_2$

We define the upsampled signal by integers $L_1 \times L_2$ as

$$x_u(n_1, n_2) \triangleq \begin{cases} x\left(\frac{n_1}{L_1}, \frac{n_2}{L_2}\right), & \text{when } L_1 \text{ divides } n_1 \text{ and } L_2 \text{ divides } n_2,^3 \\ 0, & \text{else.} \end{cases}$$

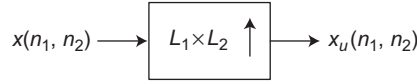
So, just as in one dimension, we define 2-D upsampling as inserting zeros for the missing samples. The system diagram for this is shown in Figure 2.3-9.

The expression for upsampling in the Fourier domain can be found as

$$\begin{aligned} X_u(\omega_1, \omega_2) &= \sum_{\text{all } n_1, n_2} x_u(n_1, n_2) \exp -j(\omega_1 n_1 + \omega_2 n_2) \\ &= \sum_{n_1=L_1 k_1, n_2=L_2 k_2} x_u(L_1 k_1, L_2 k_2) \exp -j(\omega_1 L_1 k_1 + \omega_2 L_2 k_2) \\ &= \sum_{\text{all } k_1, k_2} x_u(L_1 k_1, L_2 k_2) \exp -j(\omega_1 L_1 k_1 + \omega_2 L_2 k_2) \\ &= X(L_1 \omega_1, L_2 \omega_2). \end{aligned}$$

We note that this expression is what we would expect from the 1-D case and that no aliasing occurs due to upsampling, although there are spectral “images” appearing.

³• L divides n means just that n/L is an integer.

**FIGURE 2.3–9**

Upsample system element.

We see that upsampling effectively shrinks the frequency scale in both dimensions by the corresponding upsampling factors. Since the Fourier transform is $2\pi \times 2\pi$ periodic, this brings in many spectral repeats. Note that there is no overlap of components, though, and these repeats can therefore be removed by a lowpass filter with bandwidth $[-\pi/L_1, +\pi/L_1] \times [-\pi/L_2, +\pi/L_2]$, which leads to ideal interpolation.

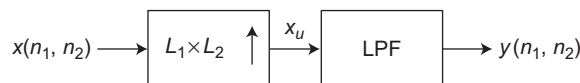
Ideal Interpolation

By ideal interpolation, we mean that interpolation that will yield the same samples as though we sampled a corresponding continuous-space function at the higher sample rate. Using the rectangular sampling reconstruction formula (2.1–5), we can evaluate at $(t_1, t_2) = (n_1 T_1/L_1, n_2 T_2/L_2)$ to obtain

$$\begin{aligned}
 x_c\left(\frac{n_1 T_1}{L_1}, \frac{n_2 T_2}{L_2}\right) &= \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) \frac{\sin \frac{\pi}{T_1} \left(\frac{n_1 T_1}{L_1} - k_1 T_1\right)}{\frac{\pi}{T_1} \left(\frac{n_1 T_1}{L_1} - k_1 T_1\right)} \frac{\sin \frac{\pi}{T_2} \left(\frac{n_2 T_2}{L_2} - k_2 T_2\right)}{\frac{\pi}{T_2} \left(\frac{n_2 T_2}{L_2} - k_2 T_2\right)} \\
 &= \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) \frac{\sin \pi \left(\frac{n_1}{L_1} - k_1\right)}{\pi \left(\frac{n_1}{L_1} - k_1\right)} \frac{\sin \pi \left(\frac{n_2}{L_2} - k_2\right)}{\pi \left(\frac{n_2}{L_2} - k_2\right)},
 \end{aligned}$$

which begins to look similar to a convolution. To achieve a convolution exactly, though, we can introduce the upsampled function x_u and write the ideal interpolation as

$$\begin{aligned}
 y &\triangleq x_c\left(\frac{n_1 T_1}{L_1}, \frac{n_2 T_2}{L_2}\right) \\
 &= \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) \frac{\sin \pi \left(\frac{n_1}{L_1} - k_1\right)}{\pi \left(\frac{n_1}{L_1} - k_1\right)} \frac{\sin \pi \left(\frac{n_2}{L_2} - k_2\right)}{\pi \left(\frac{n_2}{L_2} - k_2\right)} \\
 &= \sum_{k'_1, k'_2 = \text{multiples of } L_1, L_2}^{+\infty} x\left(\frac{k'_1}{L_1}, \frac{k'_2}{L_2}\right) \frac{\sin \pi \left(\frac{n_1 - k'_1}{L_1}\right)}{\pi \left(\frac{n_1 - k'_1}{L_1}\right)} \frac{\sin \pi \left(\frac{n_2 - k'_2}{L_2}\right)}{\pi \left(\frac{n_2 - k'_2}{L_2}\right)} \\
 &= \sum_{\text{all } k'_1, k'_2}^{+\infty} x_u(k'_1, k'_2) \frac{\sin \pi \left(\frac{n_1 - k'_1}{L_1}\right)}{\pi \left(\frac{n_1 - k'_1}{L_1}\right)} \frac{\sin \pi \left(\frac{n_2 - k'_2}{L_2}\right)}{\pi \left(\frac{n_2 - k'_2}{L_2}\right)},
 \end{aligned}$$

**FIGURE 2.3–10**

System diagram for ideal interpolation by integer factor $L_1 \times L_2$.

which is just the 2-D convolution of the upsampled signal x_u and an ideal impulse response h ,

$$h(n_1, n_2) = \frac{\sin \pi \left(\frac{n_1}{L_1} \right)}{\pi \left(\frac{n_1}{L_1} \right)} \frac{\sin \pi \left(\frac{n_2}{L_2} \right)}{\pi \left(\frac{n_2}{L_2} \right)},$$

which corresponds to an ideal rectangular lowpass filter with bandwidth $[-\pi/L_1, +\pi/L_1] \times [-\pi/L_2, +\pi/L_2]$ and passband gain $= L_1 L_2$, with the system diagram shown in Figure 2.3–10.

Example 2.3–3: Oversampling Camera

Image and video cameras only have the optical system in front of a charge-coupled device (CCD) or complementary metal oxide semiconductor (CMOS) sensor to do the anti-aliasing filtering. As such, there is often alias energy evident in the image frame, especially in the lower sample density video case. So-called *oversampling* camera chips have been introduced, which first capture the digital image at a high resolution and then do digital filtering combined with downsampling to produce the lower resolution output image. Since aliasing is generally confined to the highest spatial frequencies, the oversampled image sensor can result in a significant reduction in aliasing error.

2.4 SAMPLE-RATE CHANGE—GENERAL CASE

Just as 2-D sampling is not restricted to rectangular or Cartesian schemes, so also decimation and interpolation can be accomplished more generally in two dimensions. The 2-D grid is called a *lattice*. When we subsample a lattice, we create a sublattice—i.e., a lattice contained in the original one. If the original data came from sampling a continuous-space curve, the overall effect is then the same as sampling the continuous-space function with the sublattice. General upsampling can be viewed similarly, but we wish to go to a superlattice—i.e., one for which the given lattice is a sublattice. Here, we assume that we already have data, however obtained, and wish to either subsample it or to interpolate it. Applications occur in various areas of image and video processing, with one being the conversion of sampling lattices.

Some images are acquired from sensors on diamond sampling grids and must be upsampled to a Cartesian lattice for display on common image displays. Another application is to the problem of conversion between interlaced and progressive video by 2-D filtering in the vertical-time domain.

General Downsampling

Let the 2×2 matrix \mathbf{M} be nonsingular and contain only integer values. Then, given a discrete-space signal $x(\mathbf{n})$, where we have indicated position with the column vector \mathbf{n} , i.e., $(n_1, n_2)^T \triangleq \mathbf{n}$, a decimated signal $x_d(\mathbf{n})$ can be obtained as follows:

$$x_d(\mathbf{n}) \triangleq x(\mathbf{M}\mathbf{n}),$$

where we call \mathbf{M} the *decimation matrix*.

If we decompose \mathbf{M} into its two column vectors as

$$\mathbf{M} = [\mathbf{m}_1 \ \mathbf{m}_2],$$

then we see that we are subsampling at the locations

$$n_1\mathbf{m}_1 + n_2\mathbf{m}_2,$$

and thus we can say that these two vectors \mathbf{m}_1 and \mathbf{m}_2 are the basis vectors of the sublattice *generated by* \mathbf{M} .

Example 2.4–1: Diamond Sublattice

If we choose to subsample with the decimation matrix

$$\mathbf{M} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \quad (2.4-1)$$

then we get the sublattice consisting of all the points

$$n_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} + n_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

a portion of which is shown in Figure 2.4–1. We see the generated sublattice (i.e., the retained sample locations denoted by large filled-in circles). The left-out points are denoted by small filled-in circles. Thus the original lattice consists of all the filled-in circles. Note that we also indicate the various multiples n_1 and n_2 via the plotted axes, which then become the coordinate axes of the subsampled signal.

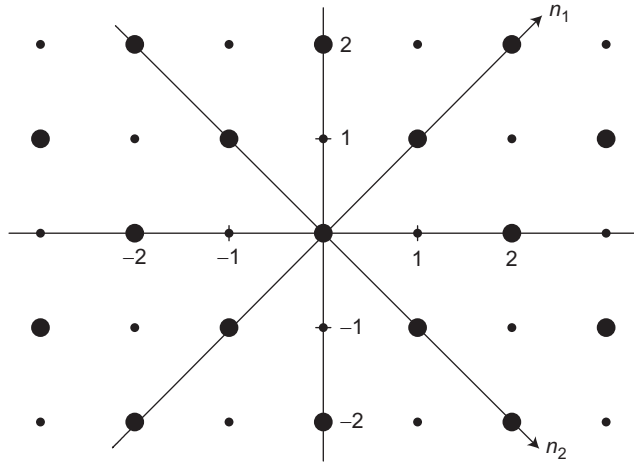
**FIGURE 2.4-1**

Illustration of portion of sublattice generated by diamond subsampling. Large filled-in circles are sublattice. Large and small filled-in circles together are the original lattice.

The corresponding result in frequency, derived by [4], is

$$X_d(\omega) = \frac{1}{|\det \mathbf{M}|} \left\{ \sum_{\text{certain } \mathbf{k}} X[\mathbf{M}^{-T}(\omega - 2\pi \mathbf{k})] \right\}, \quad (2.4-2)$$

with $\omega = (\omega_1, \omega_2)^T$, which is seen to be very close to [Example 2.2-4](#) for the general sampling case, with the decimation matrix \mathbf{M} substituted for the sampling matrix \mathbf{V} of [Section 2.2](#). Also we use X in place of the continuous-space Fourier transform X_c used there. While in (2.2-4), the sum is over all \mathbf{k} ; here, the aliasing sum is only over the additional alias points introduced by the decimation matrix \mathbf{M} . For example, looking at [Figure 2.4-1](#), we see that the small filled-in circles constitute an equivalent lattice that has been left behind by the chosen subsampling. There is a shift of $(1, 0)^T$ in this example. The derivation of (2.4-2) is considered in end-of-chapter problem 16.

Example 2.4-2: Effect of Diamond Subsampling on Frequency

Since we have the generator matrix (2.4-1) from (2.4-1), we calculate its determinant as $\det \mathbf{M} = 2$ and the transposed inverse as

$$\mathbf{M}^{-T} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

Then substituting into (2.4-2), we sum over $\mathbf{k} = (0,0)^T$ and $\mathbf{k} = (1,0)^T$ to obtain

$$\begin{aligned} Y(\omega) &= \frac{1}{2}X\left(\frac{\omega_1}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \frac{\omega_2}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) + \frac{1}{2}X\left(\frac{(\omega_1 - 2\pi)}{2}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \frac{\omega_2}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) \\ &= \frac{1}{2}X\left(\frac{\omega_1 + \omega_2}{2}, \frac{-\omega_1 + \omega_2}{2}\right) + \frac{1}{2}X\left(\frac{(\omega_1 - 2\pi)}{2} + \frac{\omega_2}{2}, \frac{-(\omega_1 - 2\pi)}{2} + \frac{\omega_2}{2}\right), \\ &= \frac{1}{2}X\left(\frac{\omega_1 + \omega_2}{2}, \frac{-\omega_1 + \omega_2}{2}\right) + \frac{1}{2}X\left(\frac{\omega_1 + \omega_2}{2} - \pi, \frac{-\omega_1 + \omega_2}{2} + \pi\right). \end{aligned}$$

We can interpret the first term as a diamond-shaped baseband and the second term as the single high-frequency aliased component. This alias signal comes from the fullband $[-\pi, +\pi]^2$ signal outside the diamond, with corners at $(0, \pi)$, $(\pi, 0)$, $(-\pi, 0)$, and $(0, -\pi)$. ■

CONCLUSIONS

This chapter has focused on how sampling theory extends to two dimensions. We first treated rectangular sampling and looked into various analog frequency supports that permit perfect reconstruction (i.e., an alias-free discrete-space representation). Then we turned to general but regular sampling patterns and focused on the commonly used hexagonal and diamond-shape patterns. Finally, we looked at sample rate change for the common rectangular sampled case and also briefly looked at the general regular subsampling problem in two dimensions.

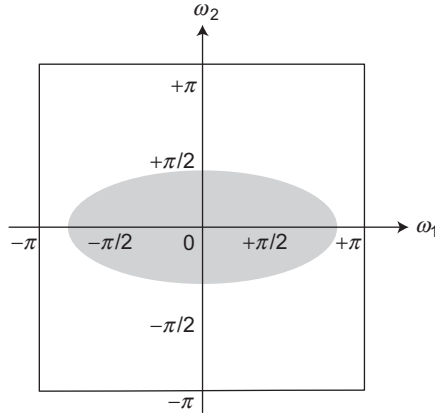
PROBLEMS

1. A certain continuous-space signal s_c is given as

$$s_c(x_1, x_2) = 100 + 20 \cos 6\pi x_1 + 40 \sin(10\pi x_1 + 6\pi x_2),$$

$$\text{for } -\infty < x_1, x_2 < +\infty.$$

- (a) What is the rectangular bandwidth of this signal? Express your answer in radians using cutoff variables Ω_{c1}, Ω_{c2} .
 - (b) Sample the function s_c at sample spacing $(\Delta_1, \Delta_2) = (0.05, 0.10)$ to obtain discrete-space signal $s(n_1, n_2) \triangleq s_c(n_1 \Delta_1, n_2 \Delta_2)$. Write an expression for s . Has overlap aliasing error occurred? Why or why not?
 - (c) Find and plot the Fourier transform $S \triangleq \text{FT}\{s\}$.
2. Denote the (discrete-space) Fourier transform as $X(\omega_1, \omega_2)$, and assume the corresponding signal $x(n_1, n_2)$ resulted from rectangularly sampling with horizontal spacing $T_1 = 10^{-3}$ meters and vertical spacing $T_2 = 10^{-4}$ meters. Assuming no aliasing, express the continuous-space Fourier transform $X_c(\Omega_1, \Omega_2)$ in

**FIGURE 2.P-1**

Ideal lowpass filter with elliptical support in the frequency domain, with cutoff frequencies ω_{c1} and ω_{c2} .

terms of the Fourier transform $X(\omega_1, \omega_2)$. Also provide a labeled sketch of X_c based on the (discrete-space) Fourier transform X shown in Figure 2.P-1.

3. A rectangularly bandlimited, continuous-space signal $s_c(x, y)$ is sampled at sample spacing T_1 and T_2 , respectively, which is sufficient to avoid spatial frequency aliasing. The resulting discrete-space signal (image) is $s(n_1, n_2)$. It is desired to process this signal to obtain what would have been obtained had the sample spacing been halved—i.e., using $T_1/2$ and $T_2/2$, respectively. What should the ideal filter impulse response $h(n_1, n_2)$ be? Call the low-rate signal s_L and the high-rate signal s_H .
4. In the reconstruction formula (2.1-4), show directly that the interpolation functions provide the correct result by evaluating (2.1-4) at the sample locations $t_1 = n_1 T_1$, $t_2 = n_2 T_2$ for the critical sampling case.
5. It is known that the continuous time 1-D signal $x_c(t) = \exp -\alpha t^2$ has Fourier transform $X_c(\Omega) = \sqrt{\frac{\alpha}{\pi}} \exp -\frac{1}{4\alpha} \Omega^2$, where $\alpha > 0$. Next, we sample $x(t)$ at times $t = n$ to obtain a discrete-time signal $x(n) = x_c(n) = \exp -\alpha n^2$.
 - (a) Write the Fourier transform $X(\omega)$ of $x(n)$ in terms of the parameter α . This should be an aliased sum involving X_c . Use sum index k .
 - (b) Find an upper bound on α so that the $k = \pm 1$ aliased terms in $X(\omega)$ are no larger than $10^{-3} \sqrt{\frac{\alpha}{\pi}}$.
 - (c) Consider the 2-D signal $x(n_1, n_2) = \exp -\alpha(n_1^2 + n_2^2)$, and repeat part (a), finding now $X(\omega_1, \omega_2)$. Use aliased sum index (k_1, k_2) .
 - (d) Find an upper bound on α so that the $(k_1, k_2) = (\pm 1, 0)$ and $(0, \pm 1)$ aliased terms in $X(\omega_1, \omega_2)$ are no larger than $10^{-3} \frac{\alpha}{\pi}$.

6. An ideal lowpass filter for upsampling is found in [Section 2.3](#):

$$h(n_1, n_2) = \frac{\sin \pi \left(\frac{n_1}{L_1} \right) \sin \pi \left(\frac{n_2}{L_2} \right)}{\pi \left(\frac{n_1}{L_1} \right) \pi \left(\frac{n_2}{L_2} \right)}.$$

Find the Fourier transform of this filter and determine its passband support and passband gain.

7. If we use 2×2 ideal interpolation for the input signal $x(n_1, n_2) = \sin(2\pi n_1/16 + 4\pi n_2/16)$, then the ideal interpolation output should be $\sin(2\pi n_1/32 + 4\pi n_2/32)$. Show that the system of [Figure 2.3–10](#) achieves this by checking the Fourier transform of the input x , the Fourier transform of the upsampled signal x_u , and the effect of the ideal lowpass filter, as determined in the reconstruction equations that follow [Figure 2.3–10](#) in the text. Note that the passband gain of this filter is here given as $2 \cdot 2 = 4$.
8. Consider the 2-D signal with continuous-space Fourier transform support as shown in the gray area of [Figure 2.P–2](#):

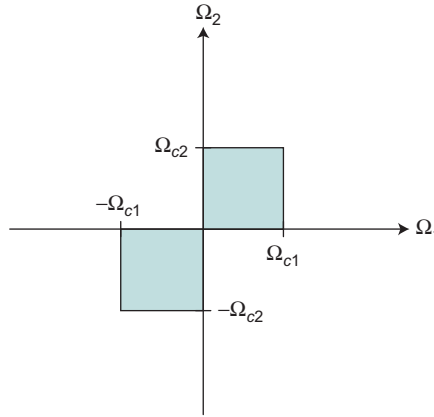


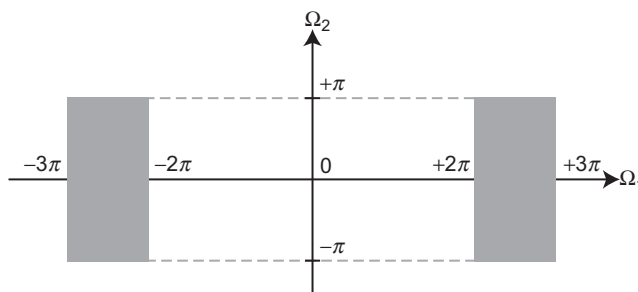
FIGURE 2.P–2

Fourier transform support indicated by gray regions.

- What is the minimum permissible rectangular sampling pattern, based on an assumption of $[-\Omega_{c1}, \Omega_{c1}] \times [-\Omega_{c2}, \Omega_{c2}]$ Fourier transform support?
- What is the minimum permissible sampling pattern for the Fourier transform support shown in the figure?

In each case, answer by specifying the sample matrix V .

9. Consider the continuous-space signal $x_c(t_1, t_2)$ with bandpass Fourier transform support as indicated by the dark gray areas in [Figure 2.P–3](#). The support of $X_c(\Omega_1, \Omega_2)$ does not include the boundaries of the dark gray boxes.

**FIGURE 2.P-3**

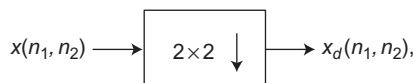
Bandpass Fourier transform support of $X_c(\Omega_1, \Omega_2)$ indicated by dark gray areas.

- (a) What is the minimal spatial sampling pattern to capture the full content of x_c so that, from these samples alone, one can reconstruct the continuous-space function x_c without any error?
 - (b) Specify the corresponding reconstruction system in detail.
10. Use Figure 2.2-4 to find the diamond-shaped unit cell in the frequency domain for sampling matrix $\mathbf{V} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. (Hint: Draw straight lines connecting the alias anchor points.) What does the midpoint of each line tell us about the boundary of the unit cell?
 11. Find the impulse response corresponding to a continuous-space ideal diamond-shaped lowpass filter, with passband equal to the frequency domain unit cell found in problem 10. This is the ideal anti-alias filter that should be used prior to such sampling. Assume $T_1 = T_2 = 1$.
 12. Show that $X_d(\omega_1, \omega_2)$ given in (2.3-5) is rectangularly periodic with period $2\pi \times 2\pi$, as it must be to be correct.
 13. Consider the 2-D signal from problem 7 of Chapter 1,

$$x(n_1, n_2) = 4 + 2 \cos \left[\frac{2\pi}{8} (n_1 + n_2) \right] + 2 \cos \left[\frac{2\pi}{8} (n_1 - n_2) \right],$$

for all $-\infty < n_1, n_2 < +\infty$.

- (a) Let x now be input to the 2×2 decimator in Figure 2.P-4

**FIGURE 2.P-4**

A 2×2 decimator.

Give a simple expression for the output $x_d(n_1, n_2)$ and its Fourier transform $X_d(\omega_1, \omega_2)$. (Hint: Consider each of the terms in x separately, starting with the constant term 4.)

(b) Check and verify that

$$X_d(\omega_1, \omega_2) = \frac{1}{4} \sum_{l_1=0}^1 \sum_{l_2=0}^1 X\left(\frac{\omega_1 - 2\pi l_1}{2}, \frac{\omega_2 - 2\pi l_2}{2}\right).$$

14. Let the signal $x(n_1, n_2)$ have Fourier transform

$$X(\omega_1, \omega_2) = \begin{cases} 1, & \text{on } [-\pi, +\pi] \times [-\frac{\pi}{2}, +\frac{\pi}{2}], \\ 0, & \text{otherwise on } [-\pi, +\pi]^2. \end{cases}$$

We then subsample 2×2 the signal x to obtain

$$x_d(n_1, n_2) = x(2n_1, 2n_2).$$

Working directly in the 2-D Fourier transform domain, find and plot the corresponding Fourier transform of the subsampled signal $X_d(\omega_1, \omega_2)$ on $[-\pi, +\pi]^2$.

15. Consider a fullband signal with circular symmetric frequency domain contour plot as sketched in Figure 2.P-5.

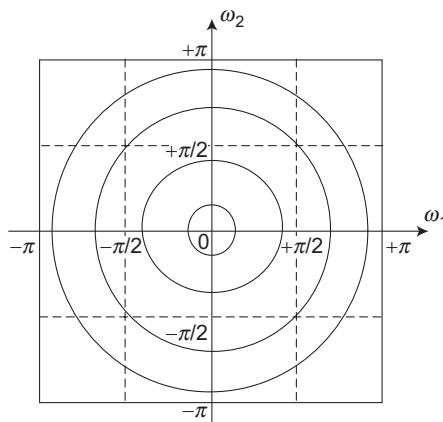


FIGURE 2.P-5

Contour plot sketch of a fullband signal.

Use a filter with frequency-domain support as shown in Figure 2.3-6 to generate the so-called HL subband. Usually this signal is then decimated 2×2 for efficiency, generating what is normally called the subband signal $x_{HL}(n_1, n_2)$. Sketch the frequency domain contour plot of this signal. (Note that there has been an inversion of higher and lower horizontal frequencies.)

16. This problem relates to the formula for frequency domain effect of general sub-sampling (2.4–2) and its derivation. Things are made much simpler by using the alternative definition of the discrete-space Fourier transform [5]:

$$\begin{aligned} X'(\omega) &\triangleq \sum_n x(\mathbf{n}) \exp(-j\omega^T \mathbf{V}\mathbf{n}) \\ &= X(\mathbf{V}^T \omega). \end{aligned}$$

With this definition, the warping evident in (2.2–4) does not occur, and we have the simpler expression for aliasing due to general sampling:

$$X'(\omega) = \frac{1}{|\det \mathbf{V}|} \left\{ \sum_{\text{all } \mathbf{k}} X_c(\omega - \mathbf{U}\mathbf{k}) \right\}.$$

Note that in the rectangular case, there is no distinction between these two, except for the familiar scaling of the analog frequency axes.

- (a) Using the alternative FT, find the frequency domain equation corresponding to direct sampling with sampling matrix $\mathbf{M}\mathbf{V}$.
- (b) Show that this same equation must correspond to decimating by \mathbf{M} after first sampling with \mathbf{V} .
- (c) Use your result in part (b) to justify (2.4–2) as expressed in terms of the alternative FT

$$X'_d(\omega) = \frac{1}{|\det \mathbf{M}|} \left\{ \sum_{\text{certain } \mathbf{k}} X(\omega - 2\pi \mathbf{M}^{-T} \mathbf{k}) \right\}.$$

- (d) Now, can you justify our using $\mathbf{k} = (0,0)^T$ and $\mathbf{k} = (1,0)^T$ in Example 2.4–2?

REFERENCES

- [1] A. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
- [2] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, 1992.
- [3] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [4] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, (see Section 12.4), Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [5] R. M. Mersereau and T. C. Speake, "The Processing of Periodically Sampled Multidimensional Signals," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-31, pp. 188–194, February 1983.

Two-Dimensional Systems and Z-Transforms

In this chapter we look at the 2-D Z-transform. It is a generalization of the 1-D Z-transform used in the analysis and synthesis of 1-D linear constant coefficient difference equation-based systems. In two and higher dimensions, the corresponding linear systems are partial difference equations. The analogous continuous parameter systems are partial differential equations. In fact, one big application of partial difference equations is in the numerical or computer solution of the partial differential equations of physics. We also look at LSI stability in terms of its Z-transform system function and present several stability conditions in terms of the zero-root locations of the system function.

3.1 LINEAR SPATIAL OR 2-D SYSTEMS

The spatial or 2-D systems we will mainly be concerned with are governed by difference equations in the two variables n_1 and n_2 . These equations can be realized by logical interconnection of multipliers, adders, and shift or delay elements via either software or hardware. For the most part, the coefficients of such equations will be constant, hence the name *linear constant coefficient difference equations* (LCCDEs). The study of 2-D or partial difference equations is much more involved than that of the corresponding 1-D LCCDEs, and much less is known about the general case. Nevertheless, many practical results have emerged, the most basic of which will be presented here. We start with the general input/output equation:

$$\sum_{(k_1, k_2) \in \mathcal{R}_a} a_{k_1, k_2} y(n_1 - k_1, n_2 - k_2) = \sum_{(k_1, k_2) \in \mathcal{R}_b} b_{k_1, k_2} x(n_1 - k_1, n_2 - k_2), \quad (3.1-1)$$

where x is the known input and y is the output to be determined. We consider the coefficients a_{k_1, k_2} and b_{k_1, k_2} to be arrays of real numbers and call b_{k_1, k_2} the *feedforward* coefficients and a_{k_1, k_2} the *feedback* coefficients. We wish to solve (3.1-1) by finding output value y for every point in a prescribed region \mathcal{R}_y given needed input values x plus output values y on the boundary of \mathcal{R}_y . We denote this boundary region somewhat imprecisely as \mathcal{R}_{bc} . The highest values of k_1 and k_2 on the left-hand side of (3.1-1) determine the order of the difference equation. In general, such equations have to be solved via matrix or iterative methods, but our main interest is *2-D filters*

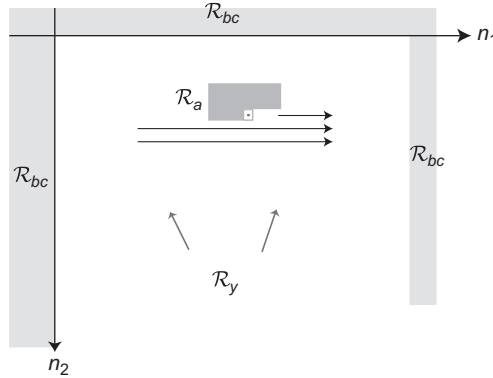


FIGURE 3.1-1

An example of the solution region of a spatial difference equation solution region using a nonsymmetric half-plane (NSHP) coefficient support \mathcal{R}_a .

for which the output y can be calculated in a recursive manner from the input x by scanning through the data points (n_1, n_2) .

Keeping only the output value $y(n_1, n_2)$ on the left-hand side of (3.1-1), assuming $a_{0,0} \neq 0$ and $(0,0) \in \mathcal{R}_a$, we can write

$$y(n_1, n_2) = - \sum_{(k_1, k_2) \in \mathcal{R}_a} a'_{k_1, k_2} y(n_1 - k_1, n_2 - k_2) + \sum_{(k_1, k_2) \in \mathcal{R}_b} b'_{k_1, k_2} x(n_1 - k_1, n_2 - k_2), \quad (3.1-2)$$

where the a'_{k_1, k_2} and b'_{k_1, k_2} are the normalized coefficients (i.e., those divided by $a_{0,0}$). Then, depending on the shape of the region \mathcal{R}_a , we may be able to calculate the solution recursively. For example, we would say that the *direction of recursion* of (3.1-2) is “downward and to the right” in Figure 3.1-1, which shows a scan proceeding left-to-right and top-to-bottom.¹ Note that the special shape of the output mask \mathcal{R}_a in Figure 3.1-1 permits such a recursion because of its property of not including any outputs that have not already been scanned and processed in the *past* (i.e., “above and to the left”).

Example 3.1-1 shows how such a recursion proceeds in the case of a simple first-order 2-D difference equation.

Example 3.1-1: Simple Difference Equation

We now consider the simple LCCDE

$$y(n_1, n_2) = x(n_1, n_2) + \frac{1}{2}[y(n_1 - 1, n_2) + y(n_1, n_2 - 1)] \quad (3.1-3)$$

¹The vertical axis is directed downward, as is common in image processing, where typically the processing proceeds from top to bottom of the image.

to be solved over the first quadrant (i.e., $\mathcal{R}_y = \{n_1 \geq 0, n_2 \geq 0\}$). In this example, we assume that the input x is everywhere zero, but that the boundary conditions given on $\mathcal{R}_{bc} = \{n_1 = -1\} \cup \{n_2 = -1\}$ are nonzero and specified by

$$\begin{aligned} y(-1, 1) &= y(-1, 2) = y(-1, 3) = 1, \\ y(-1, \text{else}) &= 0, \\ y(\text{else}, -1) &= 0. \end{aligned}$$

To calculate the solution recursively, we first determine a *scanning order*. In this case, it is the so-called *raster scan* used in video monitors: first we process the row $n_2 = 0$, starting at $n_1 = 0$ and incrementing by one each time; then we increment n_2 by one, and process the next row. With this scanning order, the difference equation (3.1–3) is seen to only use previous values of y at the “present time,” and so is recursively calculable. Proceeding to work out the solution, we obtain

	0	0	1	1	1	0	0	...
	0	0	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{7}{8}$	$\frac{7}{16}$	$\frac{7}{32}$...
$n_2 \downarrow$	0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{11}{16}$	$\frac{18}{32}$	$\frac{25}{64}$...
	0	0	$\frac{1}{8}$	$\frac{5}{16}$	$\frac{16}{32}$	$\frac{34}{64}$
	0	0	$\frac{1}{16}$...				

$\rightarrow n_1$.

In [Example 3.1–1](#) we have computed the solution to a spatial difference equation by recursively calculating out the values in a suitable scanning order, for a nonzero set of boundary “initial” conditions, but with zero input sequence. In [Example 3.1–2](#) we consider the same 2-D difference equation to be solved over the same output region, but with zero initial boundary conditions and a nonzero input. By linearity of the partial difference equation, the general case of nonzero boundaries and nonzero input follows by superposition of these two *zero-input* and *zero-state* solutions.

Example 3.1–2: Simple Difference Equation (cont’d)

We consider the simple LCCDE

$$y(n_1, n_2) = x(n_1, n_2) + \frac{1}{2}[y(n_1 - 1, n_2) + y(n_1, n_2 - 1)] \quad (3.1-4)$$

to be solved over output solution region $\mathcal{R}_y = \{n_1 \geq 0, n_2 \geq 0\}$. The boundary conditions given on $\mathcal{R}_{bc} = \{n_1 = -1\} \cup \{n_2 = -1\}$ are taken as all zeros. The input sequence is $x(n_1, n_2) = \delta(n_1, n_2)$. Starting at $(n_1, n_2) = (0, 0)$, we begin to generate the impulse response of the difference equation. Continuing the recursive calculation for the next few

columns and rows, we obtain

	0	0	0	0	0	0	...
	0	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$...
$n_2 \downarrow$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{8}$	$\frac{4}{16}$	$\frac{5}{32}$...
	0	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{6}{16}$	$\frac{10}{32}$
	0	$\frac{1}{8}$	$\frac{4}{16}$	$\frac{10}{32}$

$\rightarrow n_1$.

It turns out that this spatial impulse response has a closed-form analytic solution [1, 2],

$$y(n_1, n_2) = h(n_1, n_2) = \binom{n_1 + n_2}{n_1} 2^{-(n_1 + n_2)} u_{++}(n_1, n_2),$$

where $\binom{n_1 + n_2}{n_1}$ is the combinatorial symbol for “ $n_1 + n_2$ things taken n_1 at a time,”

$$\binom{n_1 + n_2}{n_1} = \frac{(n_1 + n_2)!}{n_1! n_2!}, \quad \text{for } n_1 \geq 0, n_2 \geq 0,$$

with $0!$ taken as 1, and where $u_{++}(n_1, n_2) = u(n_1, n_2)$ is the first quadrant unit step function. ■

Though it is usually the case that 2-D difference equations do not have a closed-form impulse response, the first-order difference equation of [Example 3.1–2](#) is one of the few exceptions. From these two examples, we can see it is possible to write the general solution to a spatial linear difference equation as a sum of a zero-input solution given rise by the boundary values plus a zero-state solution driven by the input sequence

$$y(n_1, n_2) = y_{ZI}(n_1, n_2) + y_{ZS}(n_1, n_2).$$

This generalizes the familiar 1-D systems theory result. To see this, consider a third example with both nonzero input and nonzero boundary conditions. Then note that the sum of the two solutions from these examples will solve this new problem.

In general, and depending on the output coefficient support region \mathcal{R}_a , there can be different recursive directions for (3.1–1), which we can obtain by bringing other terms to the left-hand side and recursing in other directions. For example, we can take (3.1–4) from [Example \(3.1–2\)](#) and bring $y(n_1, n_2 - 1)$ to the left-hand side to yield

$$y(n_1, n_2 - 1) = -2y(n_1, n_2) + 2x(n_1, n_2) + y(n_1 - 1, n_2),$$

or equivalently,

$$y(n_1, n_2) = -2y(n_1, n_2 + 1) + y(n_1 - 1, n_2 + 1) + 2x(n_1, n_2 + 1),$$

with direction of recursion upwards, to the right or left. So the direction in which a 2-D difference equation can be solved recursively, or recursed, depends on the support of the output or feedback coefficients (i.e., \mathcal{R}_a). For a given direction of recursion, we can calculate the output points in particular orders that are constrained

by the shape of the coefficient support region \mathcal{R}_a , resulting in an *order of computation*. In fact, there are usually several such orders of computation that are consistent with a given direction of recursion. Further, usually several output points can be calculated in parallel to speed the recursion.

Such recursive solutions are appropriate when the boundary conditions are only imposed “in the past” of the recursion—i.e., not on any points that must be calculated. In particular, with reference to Figure 3.1–1, we see no boundary conditions on the bottom of the solution region. In the more general case where there are both “initial” and “final” conditions, we can fall back on the general matrix solution for a finite region.

To solve LCCDE (3.1–1) in a finite solution region, we can use linear algebra and form a vector of the solution \mathbf{y} scanned across the region in any prespecified manner. Doing the same for the input \mathbf{x} and the boundary conditions \mathbf{y}_{bc} , we can write all the equations with one very large dimensioned vector equation,

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{y}_{bc},$$

for appropriately defined coefficient matrices \mathbf{A} and \mathbf{B} . For a 1000×1000 image, the dimension of \mathbf{y} would be 1,000,000. Here, $\mathbf{A}\mathbf{y}$ provides the terms of the equations where \mathbf{y} is on the region, and $\mathbf{B}\mathbf{y}_{bc}$ provides the terms when \mathbf{y} is on the boundary. A problem at the end of the chapter asks you to prove this fact.

If the solution region of the LCCDE is infinite, then as in the 1-D case, it is often useful to express the solution in terms of a Z-transform, which is our next topic.

3.2 Z-TRANSFORMS

Definition 3.2–1: Z-Transform

The 2-D Z-transform of a two-sided sequence $x(n_1, n_2)$ is defined as follows:

$$X(z_1, z_2) \triangleq \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}, \quad (3.2-1)$$

where $(z_1, z_2) \in \mathcal{C}^2$, the “2-D” (really 4-D) complex Cartesian product space. In general, there will be only some values of $(z_1, z_2)^T \triangleq \mathbf{z}$ for which this double sum will converge. Only *absolute convergence*,

$$\begin{aligned} & \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} |x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}| \\ &= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} |x(n_1, n_2)| |z_1|^{-n_1} |z_2|^{-n_2} < \infty, \end{aligned}$$

is considered in the theory of complex variables [3, 4], so we look for joint values of $|z_1|$ and $|z_2|$ that will yield absolute convergence. The set of \mathbf{z} for which this occurs is called

the *region of convergence*, denoted \mathcal{R}_x . In summary, a 2-D Z-transform is specified by its functional form $X(z_1, z_2)$ and its convergence region \mathcal{R}_x . ■

Similar to the 1-D case, the Z-transform is simply related to the Fourier transform, when both exist:

$$X(z_1, z_2) \Big|_{\substack{z_1=e^{j\omega_1} \\ z_2=e^{j\omega_2}}} = X(\omega_1, \omega_2),$$

with the customary abuse of notation.²

A key difference from the 1-D case is that the 2-D complex variable z exists in a 4-D space and is hard to visualize. The familiar unit circle becomes something a bit more abstract, the *unit bi-circle* in \mathcal{C}^2 [4]. The unit disk then translates over to the *unit bi-disk*, $\{|z_1|^2 + |z_2|^2 \leq 1\} \in \mathcal{C}^2$. Another key difference for two and higher dimensions is that the zeros of the Z-transform are no longer isolated. Two different *zero loci* can intersect.

Example 3.2–1: Zero Loci

Consider the following signal $x(n_1, n_2)$:

$n_2 \backslash n_1$	0	1
0	1	2
1	2	1

with assumed support $\{0, 1\} \times \{0, 1\}$. This simple four-point signal could serve, after normalization, as the impulse response of a simple directional spatial averager, giving an emphasis to structures at 45° . Proceeding to take the Z-transform, we obtain

$$X(z_1, z_2) = 1 + 2z_1^{-1} + 2z_2^{-1} + z_1^{-1}z_2^{-1}.$$

This Z-transform X is seen to exist for all \mathcal{C}^2 except for $z_1 = 0$ or $z_2 = 0$. Factoring X , we obtain

$$X(z_1, z_2) = 1 + 2z_2^{-1} + z_1^{-1}(2 + z_2^{-1}),$$

which upon equating to zero gives the zero (z_1, z_2) locus

$$z_1 = -\frac{2z_2 + 1}{z_2 + 2}, \quad \text{for } z_2 \neq -2,$$

$$z_1 = +\infty, \text{ otherwise.}$$

²To avoid confusion, when the same symbol X is being used for two different functions, we note that the Fourier transform $X(\omega_1, \omega_2)$ is a function of real variables, while the Z-transform $X(z_1, z_2)$ is a function of complex variables. A pitfall, for example $X(1, 0)$, can be avoided by simply writing either $X(\omega_1, \omega_2)|_{\substack{\omega_1=1 \\ \omega_2=0}}$ or $X(z_1, z_2)|_{\substack{z_1=1 \\ z_2=0}}$, whichever is appropriate, in cases where confusion could arise.

We notice that for each value of z_2 there is a corresponding value of z_1 for which the Z-transform X takes on the value of zero. Notice also that, with the possible exception of $z_2 = -2$, the zero locus value $z_1 = f(z_2)$ is a *continuous function* of the complex variable z_2 . This first-order 2-D system thus has one zero locus. ■

We next look at a more complicated second-order case where there are two root loci that intersect, but without being identical; therefore, we cannot just cancel the factors out. In the 1-D case that we are familiar with, the only way there can be a pole and zero at the same z location is when the numerator and denominator have a common factor. [Example 3.2–2](#) shows that this is not true in general for higher dimensions. ■

Example 3.2–2: Intersecting Zero Loci

Consider the Z-transform

$$X(z_1, z_2) = (1 + z_1)/(1 + z_1 z_2),$$

for which the zero locus is easily seen to be $(z_1, z_2) = (-1, *)$, and the pole locus is $(z_1, z_2) = (\alpha, -1/\alpha)$, where $*$ represents an arbitrary complex number and α is any nonzero complex number. These two distinct zero sets are seen to intersect at $(z_1, z_2) = (-1, 1)$. One way to visualize these root loci is *root mapping*, which we will introduce later when we study the stability of 2-D filters (see [Section 3.5](#)). ■

Next, we turn to the topic of convergence for the 2-D Z-transform. As in the 1-D case, we expect that knowledge of the region in z space where the series converges will be essential to the uniqueness of the transform, and hence to its inversion.

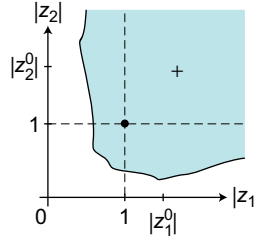
3.3 REGIONS OF CONVERGENCE

Given a 2-D Z-transform $X(z_1, z_2)$, its *region of convergence* (ROC) is given as the set of \mathbf{z} for which

$$\begin{aligned} &= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} |x(n_1, n_2)| |z_1|^{-n_1} |z_2|^{-n_2} \\ &= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} |x(n_1, n_2)| r_1^{-n_1} r_2^{-n_2} < \infty, \end{aligned} \quad (3.3-1)$$

where $r_1 \triangleq |z_1|$ and $r_2 \triangleq |z_2|$ are the moduli of the complex numbers z_1 and z_2 . The ROC can then be written in terms of such moduli values as

$$\mathcal{R}_x \triangleq \{(z_1, z_2) | |z_1| = r_1, |z_2| = r_2, \text{ and } (3.3-1) \text{ holds}\}.$$

**FIGURE 3.3–1**

The 2-D complex magnitude plane. Here, (•) denotes the unit bi-circle and (+) denotes an arbitrary point at (z_1^0, z_2^0) .

Since this specification only depends on magnitudes, we can plot ROCs in the convenient magnitude plane (Figure 3.3–1).

Example 3.3–1: Z-Transform Calculation

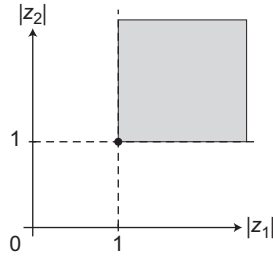
We consider the spatial, first quadrant step function

$$x(n_1, n_2) = u_{++}(n_1, n_2) = u(n_1, n_2).$$

Taking the Z-transform, we have the following from (3.2–1):

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} u(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{+\infty} \sum_{n_2=0}^{+\infty} z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{+\infty} z_1^{-n_1} \cdot \sum_{n_2=0}^{+\infty} z_2^{-n_2} \\ &= \frac{1}{1 - z_1^{-1}} \frac{1}{1 - z_2^{-1}} \quad \text{for } |z_1| > 1 \quad \text{and} \quad |z_2| > 1, \\ &= \frac{z_1}{z_1 - 1} \frac{z_2}{z_2 - 1} \quad \text{with } \mathcal{R}_x = \{|z_1| > 1, |z_2| > 1\}. \end{aligned}$$

We can plot this ROC on the complex \mathbf{z} -magnitude plane as in Figure 3.3–2. Note that we have shown the ROC as the gray region and moved it slightly outside the lines $|z_1| = 1$ and $|z_2| = 1$ in order to emphasize that this open region does not include these lines. The zero loci for this separable signal are the manifold $z_1 = 0$ and the manifold $z_2 = 0$. These two distinct loci intersect at the complex point $z_1 = z_2 = 0$. The pole loci are also two in number and occur at the manifold $z_1 = 1$ and the manifold $z_2 = 1$. We note that these two pole loci intersect at the single complex point $z_1 = z_2 = 1$.

**FIGURE 3.3-2**

The gray area illustrates the ROC for the Z-transform of the first quadrant unit step function $u(n_1, n_2) = u_{++}(n_1, n_2)$.

Next, we consider how the Z-transform changes when the unit step switches to another quadrant.

Example 3.3-2: Unit Step Function in the Fourth Quadrant

Here, we consider a unit step function that has support on the fourth quadrant. We denote it as $u_{+-}(n_1, n_2)$:

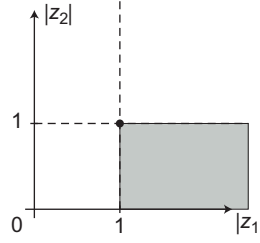
$$u_{+-}(n_1, n_2) \triangleq \begin{cases} 1, & n_1 \geq 0, n_2 \leq 0, \\ 0, & \text{else.} \end{cases}$$

So, setting $x(n_1, n_2) = u_{+-}(n_1, n_2)$, we next compute

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} u_{+-}(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{+\infty} z_1^{-n_1} \cdot \sum_{n_2=-\infty}^0 z_2^{-n_2} \\ &= \sum_{n_1=0}^{+\infty} z_1^{-n_1} \cdot \sum_{n'_2=0}^{+\infty} z_2^{n'_2} \quad \text{with } n'_2 \triangleq -n_2 \\ &= \frac{1}{1-z_1^{-1}} \frac{1}{1-z_2} \quad \text{for } |z_1| > 1 \quad \text{and} \quad |z_2| < 1, \\ &= -\frac{z_1}{z_1-1} \frac{1}{z_2-1} \quad \text{with } \mathcal{R}_x = \{|z_1| > 1, |z_2| < 1\}. \end{aligned}$$

The ROC is shown as the gray area in [Figure 3.3-3](#).

All four quarter-plane support, unit step sequences have the special property of separability. Since the Z-transform is a separable operator, this makes the calculation split into the product of two 1-D transforms in the n_1 and n_2 directions, as we have

**FIGURE 3.3-3**

The ROC (gray area) for the fourth quadrant unit step function $u_{+-}(n_1, n_2)$.

just seen. The ROC then factors into the Cartesian product of the two 1-D ROCs. We look at a more general case next.

More General Case

In general, we have the Z-transform

$$X(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)},$$

where both B and A are polynomials in coefficients of some partial difference equation,

$$B(z_1, z_2) = \sum_{n_1=-N_1}^{+N_1} \sum_{n_2=-N_2}^{+N_2} b(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \text{ and}$$

$$A(z_1, z_2) = \sum_{n_1=-N_1}^{+N_1} \sum_{n_2=-N_2}^{+N_2} a(n_1, n_2) z_1^{-n_1} z_2^{-n_2}.$$

To study the existence of this Z-transform, we focus on the denominator and rewrite A as

$$A(z_1, z_2) = z_1^{-N_1} z_2^{-N_2} \tilde{A}(z_1, z_2),$$

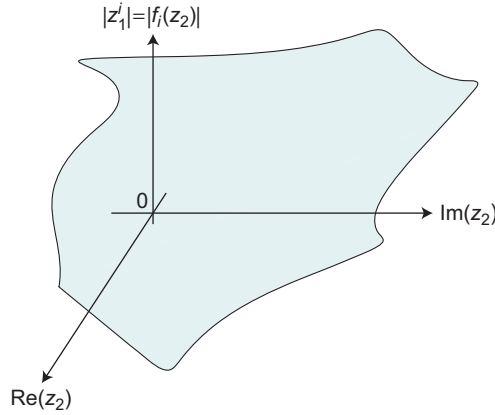
where \tilde{A} is a *strict-sense polynomial* in z_1 and z_2 (i.e., no negative powers of z_1 or z_2). Grouping together terms in z_1^n , we can write

$$\tilde{A}(z_1, z_2) = \sum_{n=0}^{\tilde{N}_1} a_n(z_2) z_1^n,$$

yielding \tilde{N}_1 poles (N_1 at most!) for each value of z_2 ,

$$z_1^i = f_i(z_2), \quad i = 1, \dots, \tilde{N}_1.$$

A sketch of such a pole surface is plotted in Figure 3.3-4. Note that we are only plotting the magnitude of one surface here, and this plot therefore does not tell the

**FIGURE 3.3-4**

Sketch of pole magnitude $|z_1^i|$ surface as a function of a point in the z_2 complex plane.

whole story. Also there are \tilde{N}_1 such sheets. Of course, there will be a similar number of zero loci or surfaces that come about from the numerator

$$\tilde{B}(z_1, z_2) = \sum_{n=0}^{\tilde{N}_1} b_n(z_2) z_1^n,$$

where $B(z_1, z_2) = z_1^{-N_1} z_2^{-N_2} \tilde{B}(z_1, z_2)$. Note that these zero surfaces can intersect the pole surfaces (as well as each other) without being identical. Thus indeterminate $\frac{0}{0}$ situations can arise that cannot be simply canceled out. One classic example is [5]

$$\frac{z_1 + z_2 - 2}{(z_1 - 1)(z_2 - 1)},$$

which evaluates to $\frac{0}{0}$ at the point $(z_1, z_2) = (1, 1)$, and yet has no cancelable factors.

3.4 SOME Z-TRANSFORM PROPERTIES

Here, we list some useful properties of the 2-D Z-transform that we will use in the sequel. Many are easy extensions of known properties of the 1-D Z-transform, but some are essentially new. In listing these properties, we introduce the symbol Z for the 2-D Z-transform operator.

Linearity property:

$$Z\{ax(n_1, n_2) + by(n_1, n_2)\} = aX(z_1, z_2) + bY(z_1, z_2), \quad \text{with ROC} = \mathcal{R}_x \cap \mathcal{R}_y.$$

Delay property:

$$Z\{x(n_1 - k_1, n_2 - k_2)\} = X(z_1, z_2) z_1^{-k_1} z_2^{-k_2}, \quad \text{with ROC} = \mathcal{R}_x.$$

Convolution property:

$$Z\{x(n_1, n_2) * y(n_1, n_2)\} = X(z_1, z_2) Y(z_1, z_2), \quad \text{with ROC} = \mathcal{R}_x \cap \mathcal{R}_y.$$

Symmetry properties:

$$Z\{x^*(n_1, n_2)\} = X^*(z_1^*, z_2^*), \quad \text{with ROC} = \mathcal{R}_x.$$

$$Z\{x(-n_1, -n_2)\} = X(z_1^{-1}, z_2^{-1}), \quad \text{with ROC} = \{(z_1, z_2) | (z_1^{-1}, z_2^{-1}) \in \mathcal{R}_x\}.$$

The proofs of these properties are very similar to the proofs in the 1-D case. [Theorem 3.4–1](#) provides a statement and proof of the very important 2-D convolution property.

Theorem 3.4–1: Z-Transform Convolution

Let $x(n_1, n_2) \longleftrightarrow X(z_1, z_2)$ and $y(n_1, n_2) \longleftrightarrow Y(z_1, z_2)$; then

$$Z\{x(n_1, n_2) * y(n_1, n_2)\} = X(z_1, z_2) Y(z_1, z_2), \quad \text{with ROC} = \mathcal{R}_x \cap \mathcal{R}_y.$$

Proof We use the vector index notation, $\mathbf{n} \triangleq (n_1, n_2)$ and $\mathbf{z} \triangleq (z_1, z_2)$, and define the output $g(\mathbf{n}) \triangleq x(\mathbf{n}) * y(\mathbf{n}) = \sum_{\mathbf{k}} x(\mathbf{k}) y(\mathbf{n} - \mathbf{k})$, and its 2-D Z-transform

$$G(\mathbf{z}) = \sum_{\mathbf{n}} g(\mathbf{n}) \mathbf{z}^{-\mathbf{n}}.$$

Note the perhaps strange symbol $\mathbf{z}^{-\mathbf{n}} \triangleq z_1^{-n_1} z_2^{-n_2}$, which is used for notational simplicity. Then we can write

$$\begin{aligned} G(\mathbf{z}) &= \sum_{\mathbf{n}} \sum_{\mathbf{k}} x(\mathbf{k}) y(\mathbf{n} - \mathbf{k}) \mathbf{z}^{-\mathbf{n}} \\ &= \sum_{\mathbf{n}} \sum_{\mathbf{k}} x(\mathbf{k}) \mathbf{z}^{-\mathbf{k}} y(\mathbf{n} - \mathbf{k}) \mathbf{z}^{-(\mathbf{n} - \mathbf{k})} \\ &= \sum_{\mathbf{k}} x(\mathbf{k}) \mathbf{z}^{-\mathbf{k}} \left[\sum_{\mathbf{n}} y(\mathbf{n} - \mathbf{k}) \mathbf{z}^{-(\mathbf{n} - \mathbf{k})} \right] \\ &= \sum_{\mathbf{k}} x(\mathbf{k}) \mathbf{z}^{-\mathbf{k}} Y(\mathbf{z}) \\ &= X(\mathbf{z}) Y(\mathbf{z}). \end{aligned}$$

For the ROC, clearly we need to have \mathbf{z} in both regions of convergence for the product to be defined; thus we set $\mathcal{R}_g = \mathcal{R}_x \cap \mathcal{R}_y$. If the intersection is null, then there is no Z-transform in this case. ■

A “new” named property for 2-D Z-transforms is given next. It is new in the sense that the corresponding 1-D result is rather insignificant.

Linear Mapping of Variables

Consider two signals $x(n_1, n_2)$ and $y(n_1, n_2)$, which are related by the so-called linear mapping of variables

$$\begin{bmatrix} n'_1 \\ n'_2 \end{bmatrix} = \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix},$$

so that $x(n_1, n_2) = y(n'_1, n'_2) = y(l_{11}n_1 + l_{12}n_2, l_{21}n_1 + l_{22}n_2)$, where the matrix components l_{ij} are all integers, as required. The following relation then holds for the corresponding Z-transforms X and Y :

$$Y(z_1, z_2) = X\left(z_1^{l_{11}} z_2^{l_{21}}, z_1^{l_{12}} z_2^{l_{22}}\right), \quad (3.4-1)$$

with convergence region

$$\mathcal{R}_y = \left\{ (z_1, z_2) \left| \left(z_1^{l_{11}} z_2^{l_{21}}, z_1^{l_{12}} z_2^{l_{22}} \right) \in \mathcal{R}_x \right. \right\}.$$

This integer mapping of variables gives a warping of the spatial points (n_1, n_2) , which is useful when we discuss stability tests and conditions for systems via their Z-transforms in a later section. A 1-D corresponding result might be the sample rate increase $x(n) = y(n') = y(2n)$. Note that we cannot, in general, go back from x to y in either one or two dimensions. However, in two dimensions, and for nontrivial cases, the inverse mapping matrix may have integer coefficients. In that case, the two signals carry the same information and are just warped versions of each other. Note that the only 1-D case then would be $x(n) = y(\pm n)$, which is almost completely constrained and not very interesting.

Example 3.4–1: Linear Mapping of Variables

In this example we use linear integer mapping of variables to map a signal from a general wedge support³ to a first quadrant support. We set

$$x(n_1, n_2) = y(n_1 + n_2, n_2),$$

with x having the support indicated in Figure 3.4–1. Now this transformation of variables also has an inverse with integer coefficients, so it is possible to also write y in terms of x :

$$y(n'_1, n'_2) = x(n'_1 - n'_2, n'_2).$$

By the relevant Z-transform property, we can say

$$Y(z_1, z_2) = X(z_1, z_1 z_2), \quad (3.4-2)$$

³By “wedge support” we mean that the signal support is first quadrant plus a wedge from the second quadrant, with the wedge indicated by a line at angle θ . For this example, $\theta = 45^\circ$.

but also, because the inverse linear mapping is also integer valued, the same property says

$$X(z_1, z_2) = Y(z_1, z_1^{-1} z_2),$$

which is alternatively easily seen by solving (3.4–2) for $X(z_1, z_2)$.

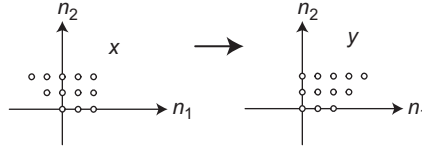


FIGURE 3.4–1

Example of linear mapping of variables.

In a later section we will develop Z-transform–based stability tests for first quadrant filters. Linear mapping of variables will be a way to extend these tests to other, more general filters—i.e., those whose denominator coefficient support is wedge shaped.

Inverse Z-Transform

The inverse Z-transform is given by the contour integral [3, 4]

$$x(n_1, n_2) = \frac{1}{(2\pi j)^2} \oint_{\mathcal{C}_1} \oint_{\mathcal{C}_2} X(z_1, z_2) z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2,$$

where the integration path $\mathcal{C}_1 \times \mathcal{C}_2$ lies completely in \mathcal{R}_x , the ROC of X , as it must. We can think of this 2-D inverse Z-transform as the concatenation of two 1-D inverse Z-transforms:

$$x(n_1, n_2) = \frac{1}{2\pi j} \oint_{\mathcal{C}_2} \left(\frac{1}{2\pi j} \oint_{\mathcal{C}_1} X(z_1, z_2) z_1^{n_1-1} dz_1 \right) z_2^{n_2-1} dz_2. \quad (3.4-3)$$

For a rational function X , the internal inverse Z-transform on the variable z_1 is straightforward albeit with poles and zeros that are a function of the other variable z_2 . For example, either partial fraction expansion or the residue method [3] could be used to evaluate the inner contour integral. Unfortunately, the second, or outer, inverse Z-transform over z_2 is often not of a rational function,⁴ and, in general, is not amenable to closed-form expression. Some simple cases can be done, though.

⁴The reason it is not generally a rational function has to do with the formulas for the roots of a polynomial. In fact, it is known that above fourth order, these 1-D polynomial roots cannot be expressed in terms of a finite number of elementary functions of the coefficients [6].

Example 3.4–2: Simple Example Resulting in a Closed-Form Solution

A simple example where the preceding integrals can be easily carried out results from the Z-transform function

$$X(z_1, z_2) = \frac{1}{1 - az_1^{-1}z_2}, \quad \text{with } \mathcal{R}_x = \left\{ |z_2| < \frac{|z_1|}{|a|} \right\},$$

where we take the case $|a| < 1$. We can illustrate this region of convergence as shown in Figure 3.4–2.

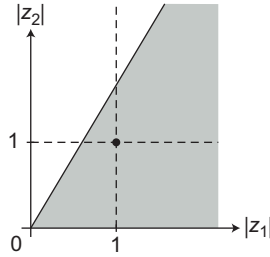
**FIGURE 3.4–2**

Illustration of ROC (shaded area) of the example Z-transform.

Proceeding with the inverse transform calculation (3.4–3), we get

$$x(n_1, n_2) = \frac{1}{2\pi j} \oint_{c_2} \left(\frac{1}{2\pi j} \oint_{c_1} \frac{1}{1 - az_1^{-1}z_2} z_1^{n_1-1} dz_1 \right) z_2^{n_2-1} dz_2.$$

Now, the inner integral corresponds to the first-order pole at $z_1 = az_2$, whose 1-D inverse Z-transform can be found using the residue method, or any other 1-D evaluation method, as

$$\frac{1}{2\pi j} \oint_{c_1} \frac{1}{1 - az_1^{-1}z_2} z_1^{n_1-1} dz_1 = (az_2)^{n_1} u(n_1).$$

Thus the overall 2-D inverse Z-transform reduces to

$$\begin{aligned} x(n_1, n_2) &= \frac{1}{2\pi j} \oint_{c_2} (az_2)^{n_1} u(n_1) z_2^{n_2-1} dz_2 \\ &= a^{n_1} u(n_1) \frac{1}{2\pi j} \oint_{c_2} z_2^{n_1+n_2-1} dz_2 \\ &= a^{n_1} u(n_1) \delta(n_1 + n_2), \end{aligned}$$

with support on the diagonal set $\{n_1 \geq 0\} \cap \{n_2 = -n_1\}$. Rewriting, this result becomes

$$x(n_1, n_2) = \begin{cases} a^{n_1}, & n_1 \geq 0 \quad \text{and} \quad n_2 = -n_1, \\ 0, & \text{else.} \end{cases}$$

Other methods to invert the 2-D Z-transform are:

- Direct long division of the polynomials
- Known series expansion
- Use of Z-transform properties on known transform pairs

Example 3.4–3: Long Division Method for Inverse Z-Transform

We illustrate the long division method with the following example. Let

$$X(z_1, z_2) = \frac{1}{1 - z_1^{-1} z_2}, \quad \text{with} \quad \mathcal{R}_x = \{|z_2| < |z_1|\}.$$

We proceed to divide the denominator into the numerator as follows:

$$\begin{array}{r} 1 + z_1^{-1} z_2 + z_1^{-2} z_2^2 + \cdots \\ 1 - z_1^{-1} z_2 \overline{\sqrt{1}} \\ \hline 1 - z_1^{-1} z_2 \\ \hline z_1^{-1} z_2 \\ \hline z_1^{-1} z_2 - z_1^{-2} z_2^2 \\ \hline z_1^{-2} z_2^2 \\ \hline z_1^{-2} z_2^2 - z_1^{-3} z_2^3 \\ \hline z_1^{-3} z_2^3 \\ \hline \vdots \end{array}$$

So

$$\frac{1}{1 - z_1^{-1} z_2} = 1 + z_1^{-1} z_2 + z_1^{-2} z_2^2 + z_1^{-3} z_2^3 + \cdots,$$

which converges for $|z_1^{-1} z_2| < 1$ (i.e., $|z_2| < |z_1|$).

The 1-D partial fraction expansion method for inverse Z-transform does not carry over to the 2-D case. This is a direct result of the nonfactorability of polynomials in more than one variable.

Example 3.4–4: 2-D Polynomials Do Not Factor

In the 1-D case, all high-order polynomials factor into first-order factors, and this property is used in a partial fraction expansion. In the multidimensional case, polynomial

factorization cannot be guaranteed anymore. In fact, most of the time it is absent, as the following example illustrates. Consider a signal x , with 3×3 support, corresponding to a 2×2 order polynomial in z_1 and z_2 . If we could factor this polynomial into first-order factors, that would be the same as representing this signal as the convolution of two signals, say a and b , of support 1×1 . We would have

$$\begin{array}{ccc} x_{02} & x_{12} & x_{22} \\ x_{01} & x_{11} & x_{21} \\ x_{00} & x_{10} & x_{20} \end{array} = \begin{array}{cc} a_{01} & a_{11} \\ a_{00} & a_{10} \end{array} \circledast \begin{array}{cc} b_{01} & b_{11} \\ b_{00} & b_{10} \end{array},$$

or in Z-transforms (polynomials), we would have

$$X(z_1, z_2) = A(z_1, z_2)B(z_1, z_2).$$

The trouble here is that a general such x has nine degrees of freedom, while the total number of unknowns in a and b is only eight. If we considered factoring a general $N \times N$ array into two factors of order $N/2 \times N/2$, the deficiency in number of unknowns (variables) would be much greater—i.e., N^2 equations versus $2(N/2)^2 = N^2/2$ unknowns! ■

The fact that multidimensional polynomials do not factor has a number of other consequences, beyond the absence of a partial fraction expansion. It means that in filter design, one must take the implementation into account at the design stage. We will see in Chapter 5 that if one wants, say, the 2-D analog of second-order factors, then one has to solve the design approximation problem with this constraint built in. But also a factored form may have larger support than the corresponding nonfactored form, thus *possibly* giving a better approximation for the same number of coefficients. So this nonfactorability is not necessarily bad. Since we can always write the preceding first-order factors as separable, then 2-D polynomial factorability down to first-order factors would lead back to a finite set of isolated poles in each complex plane.

Generally, we think of the Fourier transform as the evaluation of the Z-transform on the unit polycircle $\{|z_1| = |z_2| = 1\}$; however, this assumes the polycircle is in the region of convergence of $X(z_1, z_2)$, which is not always true. The next example demonstrates this exception.

Example 3.4–5: Comparison of Fourier Transform and Z-Transform

Consider the first quadrant unit step function $u_{++}(n_1, n_2)$. Computing its Z-transform, we earlier obtained

$$U_{++}(z_1, z_2) = \frac{1}{1 - z_1^{-1}} \frac{1}{1 - z_2^{-1}},$$

which is well behaved in its region of convergence $\mathcal{R}_x = \{|z_1| > 1, |z_2| > 1\}$. Note, however, that the corresponding Fourier transform has a problem here and needs impulses to get

by. In fact, using the separability of $u_{++}(n_1, n_2)$ and noting the 1-D Fourier transform pair,

$$u(n) \Leftrightarrow U(\omega) = \pi \delta(\omega) + \frac{1}{1 - e^{-j\omega}},$$

we obtain the 2-D Fourier transform,

$$U_{++}(\omega_1, \omega_2) = \left[\pi \delta(\omega_1) + \frac{1}{1 - e^{-j\omega_1}} \right] \left[\pi \delta(\omega_2) + \frac{1}{1 - e^{-j\omega_2}} \right].$$

If we consider the convolution of two of these step functions, this should correspond to multiplying the transforms together. For the Z-transform, there is no problem, and the region of convergence remains unchanged. For the Fourier transform, though, we would have to be able to interpret $[\delta(\omega_1)]^2$ which is not possible, as powers and products of singularity functions are unstudied and so not defined.

Conversely, a sequence that has no Z-transform but does have a Fourier transform is $\sin \omega n$, which in two dimensions becomes the plane wave $\sin(\omega_1 n_1 + \omega_2 n_2)$. Thus each transform has its own useful place. The Fourier transform is not strictly a subset of the Z-transform, because it can use impulses and other singularity functions, which are not permitted to Z-transforms. ■

We next turn to the stability problem for 2-D systems and find that the Z-transform plays a prominent role just as in one dimension. However, the resulting 2-D stability tests are much more complicated, since the zeros and poles are functions, not points, in 2-D \mathbf{z} space.

3.5 2-D FILTER STABILITY

Stability is an important concept for spatial filters, as in the 1-D case. The finite impulse response (FIR) filters are automatically stable due to the finite number of presumably finite-valued coefficients. Basically, stability means that the filter response will never get too large, if the input is bounded. For a linear filter, this means that nothing unpredictable or chaotic could be caused by extremely small perturbations in the input. A related notion is sensitivity to inaccuracies in the filter coefficients and computation, and stability is absolutely necessary to have the desired low sensitivity. Stability is also necessary so that boundary conditions (often unknown in practice) will not have a big effect on the response far from the boundary. So, stable systems are preferred for a number of practical reasons. We start with the basic definition of stability of an LSI system.

Definition 3.5–1: Stability of an LSI System

We say a 2-D LSI system with impulse response $h(n_1, n_2)$ is bounded-input bounded-output (BIBO) stable if

$$\|h\|_1 \triangleq \sum_{k_1, k_2=-\infty}^{+\infty} |h(k_1, k_2)| < \infty.$$

The resulting linear space of signals is called l_1 , and the norm $\|h\|_1$ is called the l_1 norm. Referring to this signal space, we can say that the impulse response is BIBO stable *if and only if* (iff) it is an element of the l_1 linear space; that is,

$$\text{system is stable} \iff h \in l_1. \quad (3.5-1)$$

Clearly, this means that for such a system described by convolution, the output will be uniformly bounded if the input is uniformly bounded. As we recall from Chapter 1, Section 1.1,

$$\begin{aligned} |y(n_1, n_2)| &= \left| \sum_{k_1, k_2} x(n_1 - k_1, n_2 - k_2) h(k_1, k_2) \right| \\ &\leq \max |x(k_1, k_2)| \cdot \left| \sum_{k_1, k_2} h(k_1, k_2) \right| \\ &= M \cdot \|h\|_1 \\ &< \infty, \end{aligned}$$

where $M < \infty$ is the assumed finite uniform bound on the input signal magnitude $|x|$, and the l_1 norm $\|h\|_1$ is assumed finite.

This BIBO stability condition is most easily expressed in terms of the Z-transform system function H . Because for Z-transforms convergence is really absolute convergence, the system is stable if

$$\text{the point } (z_1, z_2) = (1, 1) \in \mathcal{R}_h,$$

where \mathcal{R}_h is the ROC of $H(z_1, z_2)$. In general, for a rational system, we can write

$$H(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)},$$

where A and B are 2-D polynomials in the variables z_1 and z_2 . In one dimension, the corresponding statement would be $H(z) = B(z)/A(z)$, and stability would be determined by whether $1 \in \text{ROC of } 1/A$; that is, we could ignore the numerator's presence, assuming no pole-zero cancellations. However, in the case of two and higher dimensions, this is no longer true, and it has been shown [5] that partial pole-zero cancellations can occur in such a way that no common factor can be removed, and so that the resulting filter is stable *because* of the effect of this numerator. Since this is a somewhat delicate situation, here we look for a more robust stability, and so look at just the *poles* of the system H , which are the zeros of the denominator polynomial $A(z_1, z_2)$. For such *robust stability*, we must require $A(z_1, z_2) \neq 0$ in a region including $(z_1, z_2) = (1, 1)$, analogously to the 1-D case.

First Quadrant Support

If a 2-D digital filter has first quadrant support, then we have the following property for its Z-transform. If the Z-transform converges for some point (z_1^0, z_2^0) , then it must also converge for all $(z_1, z_2) \in \{|z_1| \geq |z_1^0|, |z_2| \geq |z_2^0|\}$ because for such (z_1, z_2) we have

$$\sum_{k_1 \geq 0, k_2 \geq 0} |h(k_1, k_2)| |z_1|^{-k_1} |z_2|^{-k_2} \leq \sum_{k_1 \geq 0, k_2 \geq 0} |h(k_1, k_2)| |z_1^0|^{-k_1} |z_2^0|^{-k_2}.$$

Thus since filters that are stable must have $(1, 1) \in \mathcal{R}_h$, first quadrant support filters that are stable must have an ROC that includes $\{|z_1| \geq 1, |z_2| \geq 1\}$, as sketched in Figure 3.5–1 as the gray region. Note that the region slightly overlaps the lines $|z_1| = 1$ and $|z_2| = 1$ in order to emphasize that these lines must be contained in the open region that is the ROC of a first quadrant support and stable filter.

We are not saying that the ROC of first quadrant support stable filters will look like that sketched in Figure 3.5–1, just that the convergence region must include the gray area.

Second Quadrant Support

If a 2-D digital filter has second quadrant support, then we have the following property for its Z-transform. If the Z-transform converges for some point (z_1^0, z_2^0) , then it must also converge for all $(z_1, z_2) \in \{|z_1| \leq |z_1^0|, |z_2| \geq |z_2^0|\}$, because for such (z_1, z_2) we have

$$\sum_{k_1 \leq 0, k_2 \geq 0} |h(k_1, k_2)| |z_1|^{-k_1} |z_2|^{-k_2} \leq \sum_{k_1 \leq 0, k_2 \geq 0} |h(k_1, k_2)| |z_1^0|^{-k_1} |z_2^0|^{-k_2}.$$

Thus since filters that are stable must have $(1, 1) \in \mathcal{R}_h$, second quadrant support filters that are stable must have an ROC that includes $\{|z_1| \leq 1, |z_2| \geq 1\}$, as sketched in Figure 3.5–2 as the gray region. Note that the region slightly overlaps the lines $|z_1| = 1$ and $|z_2| = 1$ in order to emphasize that these lines must be contained in the open region that is the ROC of a second quadrant support and stable filter.

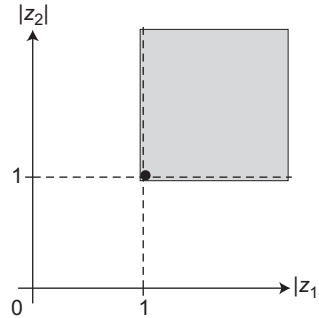


FIGURE 3.5–1

Region that must be included in the ROC of a stable first quadrant support filter.

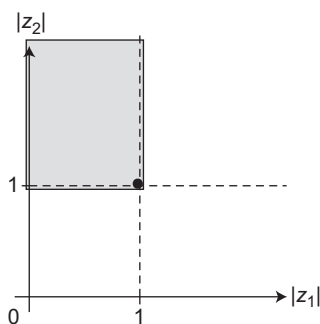


FIGURE 3.5-2

Region that must be included in the ROC of a second quadrant support stable filter.

Example 3.5-1: A Region of Convergence

Consider the spatial filter with impulse response

$$h(n_1, n_2) = \begin{cases} a^{n_1}, & n_1 \leq 0 \text{ and } n_2 = -n_1, \\ 0, & \text{else,} \end{cases}$$

with $|a| > 1$. Note that the support of this h is in the second quadrant.

Computing the Z-transform, we obtain

$$\begin{aligned} H(z_1, z_2) &= 1 + a^{-1} z_1 z_2^{-1} + a^{-2} z_1^2 z_2^{-2} + \dots \\ &= \frac{1}{1 - a^{-1} z_1 z_2^{-1}}, \quad \text{with } \mathcal{R}_h = \{|a|^{-1} |z_1| |z_2|^{-1} < 1\}. \end{aligned}$$

We can sketch this ROC as shown in Figure 3.5-3.

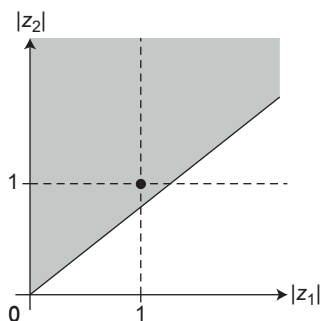


FIGURE 3.5-3

Sketch of ROC (gray area) of example Z-transform.

If a 2-D filter has support in either of the remaining two quadrants, the ROC minimal size region can easily be determined similarly. We have thus proved our first theorems on spatial filter stability.

Theorem 3.5–1: Shanks et al. [7]

A 2-D or spatial filter with first quadrant support impulse response $h(n_1, n_2)$ and rational system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is BIBO stable if $A(z_1, z_2) \neq 0$ in a region including $\{|z_1| \geq 1, |z_2| \geq 1\}$. Ignoring the effect of the numerator $B(z_1, z_2)$, this is the same as saying that $H(z_1, z_2)$ is *analytic* (i.e., $H \neq \infty$) in this region. ■

For second quadrant impulse response support, we can restate this theorem.

Theorem 3.5–2:

A 2-D or spatial filter with second quadrant support impulse response $h(n_1, n_2)$ and rational system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is BIBO stable if $A(z_1, z_2) \neq 0$ in a region including $\{|z_1| \leq 1, |z_2| \geq 1\}$. ■

Similarly, here are the theorem restatements for filters with impulse response support on either of the remaining two quadrants.

Theorem 3.5–3:

A 2-D or spatial filter with third quadrant support impulse response $h(n_1, n_2)$ and rational system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is BIBO stable if $A(z_1, z_2) \neq 0$ in a region including $\{|z_1| \leq 1, |z_2| \leq 1\}$. ■

Theorem 3.5–4:

A 2-D or spatial filter with fourth quadrant support impulse response $h(n_1, n_2)$ and rational system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is BIBO stable if $A(z_1, z_2) \neq 0$ in a region including $\{|z_1| \geq 1, |z_2| \leq 1\}$. ■

If we use the terminology ++ to refer to impulse responses with first quadrant support, +− referring to those with second quadrant support, and so forth, then all four zero-free regions for denominator polynomials A can be summarized in the diagram of [Figure 3.5–4](#). In words, we say “a ++ support filter must have ROC including $\{|z_1| \geq 1, |z_2| \geq 1\}$,” which is shown as the ++ region in this figure. A general support spatial filter can be made up from these four components as either a

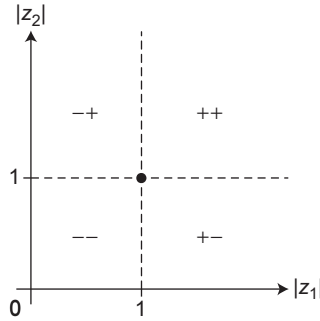
**FIGURE 3.5-4**

Illustration of necessary convergence regions for all four quarter-plane support filters.

sum or convolution product of these quarter-plane impulse responses:

$$h(n_1, n_2) = h_{++}(n_1, n_2) + h_{+-}(n_1, n_2) + h_{-+}(n_1, n_2) + h_{--}(n_1, n_2) \text{ or}$$

$$h(n_1, n_2) = h_{++}(n_1, n_2) * h_{+-}(n_1, n_2) * h_{-+}(n_1, n_2) * h_{--}(n_1, n_2).$$

Both these general representations are useful for 2-D recursive filter design, as we will see in Chapter 5.

To test for stability using the preceding theorems would be quite difficult for all but the lowest order polynomials A , since we have seen that the zero loci of 2-D functions are continuous functions in one of the complex variables, hence requiring the evaluation of an infinite number of roots. Fortunately, it is possible to simplify these theorems. We will take the case of the $++$ or first quadrant support filter here as a prototype.

Theorem 3.5-5: Simplified $++$ Test

A $++$ filter with first quadrant support impulse response $h(n_1, n_2)$ and rational system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is BIBO stable if

- (a) $A(e^{j\omega_1}, z_2) \neq 0$ in a region including $\{\text{all } \omega_1, |z_2| \geq 1\}$
- (b) $A(z_1, e^{j\omega_2}) \neq 0$ in a region including $\{|z_1| \geq 1, \text{all } \omega_2\}$

Proof We can construct a proof by relying on Figure 3.5-5. Here, the \times represent known locations of the roots when either $|z_1| = 1$ or $|z_2| = 1$. Now, since it is known in mathematics that the roots must be continuous functions of their coefficients [6], and since the coefficients, in turn, are simple polynomials in the other variable, it follows that each of these roots must be a continuous function of either z_1 or z_2 . Now, as $|z_1| \nearrow$, beyond 1, the roots cannot cross the line $|z_2| = 1$ because of condition b. Also, as $|z_2| \nearrow$, beyond 1, the roots cannot cross the line $|z_1| = 1$ because of condition a. We thus conclude that the region $\{|z_1| \geq 1, |z_2| \geq 1\}$ will be zero free by virtue of conditions a and b as was to be shown.

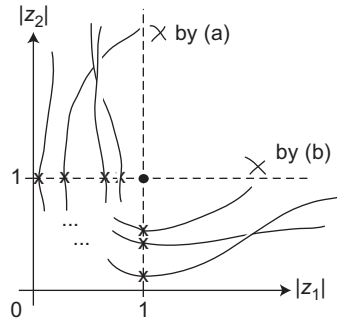
**FIGURE 3.5-5**

Figure used in proof of [Theorem 3.5-5](#). ■

This theorem has given us considerable complexity reduction, in that the 3-D conditions a and b can be tested with much less work than can the original 4-D C^2 condition. To test condition a of the theorem, we must find the roots in z_2 of the indicated polynomial, whose coefficients are a function of the scalar variable ω_1 , with the corresponding test for theorem condition b .

Root Maps

We can gain more insight into the previous theorem by using the technique of root mapping. The idea is to fix the magnitude of one complex variable, say $|z_1| = 1$, and then find the location of the root locus in the z_2 plane as the phase of z_1 varies. Looking at condition a of [Theorem 3.5-5](#), we see that as ω_1 varies from $-\pi$ to $+\pi$, the roots in the z_2 plane must stay inside the unit circle there, in order for the filter to be stable. In general, there will be N such roots or root maps. Because the coefficients of the z_2 polynomial are continuous functions of z_1 , as mentioned previously, it follows from a theorem in algebra that the root maps will be continuous functions [6]. Since the coefficients are periodic functions of ω_1 , the root maps will additionally be closed curves in the z_2 plane. This is illustrated in [Figure 3.5-6](#). Thus a test for stability based on root maps can be constructed from [Theorem 3.5-5](#) by considering both sets of root maps corresponding to part (a) and part (b) and showing that all the root maps stay inside the unit circle in the target z plane.

Two further Z-transform stability theorem simplifications have been discovered as indicated in the following two stability theorems.

Theorem 3.5-6: Huang [8]

The stability of a first quadrant or ++ quarter-plane filter of the form $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is assured by the following two conditions:

- (a) $A(e^{j\omega_1}, z_2) \neq 0$ in a region including $\{\text{all } \omega_1, |z_2| \geq 1\}$
- (b) $A(z_1, 1) \neq 0$ for all $|z_1| \geq 1$

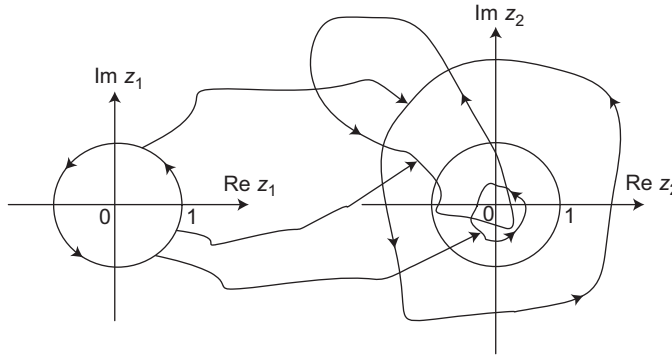
**FIGURE 3.5-6**

Illustration of root map of condition (a) of the previous [Theorem 3.5-5](#).

Proof By condition *a*, the root maps in the z_2 plane are all inside the unit circle. But also, by condition *a*, none of the root maps in the z_1 plane can cross the unit circle. Then condition *b* states that one point on these z_1 plane root maps is inside the unit circle (i.e., the point corresponding to $\omega_2 = 0$). By the continuity of the root map, the whole z_1 plane root map must lie inside the unit circle. Thus by [Theorem 3.5-5](#), the filter is stable. ■

A final simplification results in the next theorem.

Theorem 3.5-7: DeCarlo–Strintzis [9]

The stability of a first quadrant or ++ quarter-plane filter of the form $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ is assured by the following three conditions:

- (a) $A(e^{j\omega_1}, e^{j\omega_2}) \neq 0$ for all ω_1, ω_2
- (b) $A(z_1, 1) \neq 0$ for all $|z_1| \geq 1$
- (c) $A(1, z_2) \neq 0$ for all $|z_2| \geq 1$

Proof Here, condition *a* tells us that no root maps cross the respective unit circles. So, we know that the root maps are either completely outside the unit circles or completely inside them. Conditions *b* and *c* tell us that there is one point on each set of root maps that is inside the unit circles. We thus conclude that all root maps are inside the unit circles in their respective z planes. ■

A simple example of the use of these stability theorems in stability tests follows.

Example 3.5-2: Filter Stability Test

Consider the spatial filter with impulse response support on the first quadrant and system function

$$H(z_1, z_2) = \frac{1}{1 - az_1^{-1} + bz_2^{-1}},$$

so that we have $A(z_1, z_2) = 1 - az_1^{-1} + bz_2^{-1}$. By [Theorem 3.5–1](#), we must have $A(z_1, z_2) \neq 0$ for $\{|z_1| \geq 1, |z_2| \geq 1\}$. Thus all the roots $z_2 = f(z_1)$ must satisfy $|z_2| = |f(z_1)| < 1$ for all $|z_1| > 1$. In this case, we have, by setting $A(z_1, z_2) = 0$, that

$$z_2 = f(z_1) = \frac{b}{1 - az_1^{-1}}.$$

Consider $|z_1| > 1$; assuming that we must have $|a| < 1$, then

$$|z_2| = \frac{|b|}{|1 - az_1^{-1}|} \leq \frac{|b|}{1 - |a|},$$

with the maximum value being achieved for some phase angle of z_1 . Hence, for BIBO stability of this ++ filter, we need

$$\frac{|b|}{1 - |a|} < 1 \text{ or equivalently } |a| + |b| < 1.$$

The last detail now is to deal with the assumption that $|a| < 1$. Actually, it is easy to see that this is necessary, by setting $|z_2| = \infty$ and noting the root at $z_1 = a$, which must be inside the unit circle. ■

Stability Criteria for NSHP Support Filters

The ++ quarter-plane support filter is not the most general one that can follow a raster scan (horizontal across, and then down) and produce its output recursively, or what is called a recursively computable set of equations. We saw in [Example 3.4–1](#) a wedge support polynomial that can be used in place of the quarter-plane $A(z_1, z_2)$ of the previous subsection. The most general wedge support, which is recursively computable in this sense, would have support restricted to a *nonsymmetric half-plane* (NSHP), defined as

$$\text{a NSHP region} \triangleq \{n_1 \geq 0, n_2 \geq 0\} \cup \{n_1 < 0, n_2 > 0\},$$

and illustrated in [Figure 3.5–7](#). With reference to this figure, we can see that an NSHP filter makes wider use of the previously computed outputs, assuming a conventional raster scan, and hence is expected to have some advantages. We can see that this NSHP filter is a generalization of a first quadrant filter, which includes some points from a neighboring quadrant, in this case the second. Extending our notation, we can call such a filter a \oplus -NSHP filter, with other types being denoted \ominus +, $+\ominus$, etc. We next present a stability test for this type of spatial recursive filter.

Theorem 3.5–8: (NSHP Filter Stability [10])

A \oplus -NSHP support spatial filter is stable in the BIBO sense if its system function $H(z_1, z_2) = B(z_1, z_2)/A(z_1, z_2)$ satisfies:

- (a) $H(z_1, \infty)$ is analytic, i.e., free of singularities, on $\{|z_1| \geq 1\}$
- (b) $H(e^{+j\omega_1}, z_2)$ is analytic on $\{|z_2| \geq 1\}$, for all $\omega_1 \in [-\pi, +\pi]$

■

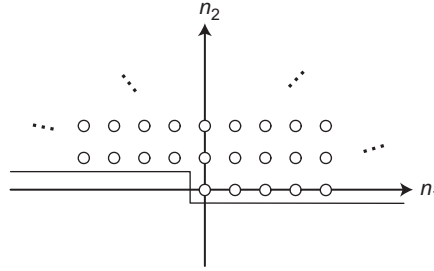


FIGURE 3.5-7

An illustration of NSHP coefficient array support.

Discussion Ignoring possible effects of the numerator on stability, condition a states that $A(z_1, \infty) \neq 0$ on $\{|z_1| \geq 1\}$, and condition b is equivalently the condition $A(e^{+j\omega_1}, z_2) \neq 0$ on $\{|z_2| \geq 1\}$ for all $\omega_1 \in [-\pi, +\pi]$, which we have seen several times before. To see what the stability region should be for a $\oplus+$ NSHP filter, we must realize that now we can no longer assume that $n_1 \geq 0$, so if the Z-transform converges for some point (z_1^0, z_2^0) , then it must also converge for all $(z_1, z_2) \in \{|z_1| = |z_1^0|, |z_2| \geq |z_2^0|\}$. By filter stability, we know that $h \in l_1$ so that the Z-transform H must converge for some region including $\{|z_1| = |z_2| = 1\}$; thus the ROC for H a $\oplus+$ NSHP filter must only include $\{|z_1| = 1, |z_2| \geq 1\}$. To proceed further, we realize that for any $\oplus+$ NSHP filter $H = 1/A$, we can write

$$A(z_1, z_2) = A(z_1, \infty)A_1(z_1, z_2),$$

with $A_1(z_1, z_2) \triangleq A(z_1, z_2)/A(z_1, \infty)$. Then the factor A_1 will not contain any coefficient terms $a_1(n_1, n_2)$ on the current line. As a result its stability can be completely described by the requirement that $A_1(e^{+j\omega_1}, z_2) \neq 0$ on $\{|z_2| \geq 1\}$ for all $\omega_1 \in [-\pi, +\pi]$.⁵ Similarly, to have the first factor stable, in the $+n_1$ direction, we need $A(z_1, \infty) \neq 0$ on $\{|z_1| \geq 1\}$. Given both conditions a and b then, the $\oplus+$ NSHP filter will be BIBO stable.

This stability test can be used on first quadrant quarter-plane filters since they are a subset of the NSHP support class. If we compare to Huang's theorem (Theorem 3.5-6), for example, we see that condition b here is like condition a there, but the 1-D test (condition a here) is slightly simpler than the 1-D test (condition b there). Here, we just take the 1-D coefficient array on the horizontal axis for this test, while in Theorem 3.5-6, we must add the coefficients in vertical columns first.

⁵Filters with denominators solely of form A_1 are called *symmetric half-plane* (SHP).

Example 3.5–3: Stability Test of ++ Filter Using NSHP Test

Consider again the spatial filter with impulse response support on the first quadrant (i.e., a ++ support filter) and system function

$$H(z_1, z_2) = \frac{1}{1 - az_1^{-1} + bz_2^{-1}},$$

so that we have $A(z_1, z_2) = 1 - az_1^{-1} + bz_2^{-1}$. Since it is a subclass of the \oplus +NSHP filters, we can apply the theorem just presented. First, we test condition *a*: $A(z_1, \infty) = 1 - az_1^{-1} = 0$ implies $z_1 = a$, and hence we need $|a| < 1$. Then, testing condition *b*, we have: $A(e^{+j\omega_1}, z_2) = 1 - ae^{-j\omega_1} - bz_2^{-1} = 0$, which implies that

$$z_2 = \frac{b}{1 - ae^{-j\omega_1}}.$$

Since we need $|z_2| < 1$, we get the requirement $|a| + |b| < 1$, just as before. ■

The next example uses the NSHP stability test on an NSHP filter, where it is needed.

Example 3.5–4: Test of NSHP Filter

Consider an impulse response with \oplus + NSHP support, given as

$$H(z_1, z_2) = \frac{1}{1 - az_1^{-1} + bz_1z_2^{-1}},$$

where we see that the recursive term in the previous line is now “above and to the right,” instead of “above,” as in a ++ quarter-plane support filter. First, we test condition *a*: $A(z_1, \infty) = 1 - az_1^{-1} = 0$ implies $z_1 = a$, and hence we need $|a| < 1$. Then, testing condition *b*, we have: $A(e^{+j\omega_1}, z_2) = 1 - ae^{-j\omega_1} - be^{j\omega_1}z_2^{-1} = 0$, which implies that

$$z_2 = \frac{be^{j\omega_1}}{1 - ae^{-j\omega_1}}.$$

Since we need $|z_2| < 1$, we get the requirement $|a| + |b| < 1$, just as before. ■

CONCLUSIONS

This chapter has looked at how the Z-transform generalizes to two dimensions. We have also looked at spatial difference equations and their stability in terms of the

Z-transform. The main difference with the 1-D case is that 2-D polynomials do not generally factor into first, or even lower order factors. As a consequence, we found that poles and zeros of Z-transforms were not isolated and have turned out to be surfaces in the multidimensional complex space. Filter stability tests are much more complicated, although we have managed some simplifications that are computationally effective given today's computer power. Later, when we study filter design, we will incorporate the structure and a stability constraint into the formulation. We also introduced filters with the more general nonsymmetric half-plane (NSHP) support and briefly investigated their stability behavior in terms of Z-transforms.

PROBLEMS

1. Find the 2-D Z-transform and region of convergence (ROC) of each of the following:
 - (a) $u_{++}(n_1, n_2)$
 - (b) $\rho^{(n_1+n_2)}, |\rho| < 1$
 - (c) $b^{(n_1+2n_2)}u(n_1, n_2)$
 - (d) $u_{-+}(n_1, n_2)$

2. Show that

$$y(n_1, n_2) = \binom{n_1 + n_2}{n_1} a^{n_1} n^{n_2} u_{++}(n_1, n_2)$$

satisfies the spatial difference equation

$$y(n_1, n_2) = ay(n_1 - 1, n_2) + by(n_1, n_2 - 1) + \delta(n_1, n_2)$$

over the region $n_1 \geq 0, n_2 \geq 0$. Assume initial rest—i.e., all boundary conditions on top- and left-hand sides are zero.

3. For the impulse response found in [Example 3.1–2](#), use Stirling's formula⁶ for the factorial to estimate whether this difference equation can be a stable filter or not.
4. In [Section 3.1](#), it is stated that the solution of a set of LCCDEs over a region can be written as

$$y(n_1, n_2) = y_{ZI}(n_1, n_2) + y_{ZS}(n_1, n_2),$$

where y_{ZI} is the solution to the boundary conditions with the input x set to zero, and y_{ZS} is the solution to the input x subject to zero boundary conditions. Show why this is true.

⁶A simple version of Stirling's formula (Stirling, 1730) is as follows: $n! \approx \sqrt{2\pi n} n^n e^{-n}$.

5. Find the Z-transform system function $H(z_1, z_2)$ of the LCCDE

$$y(n_1, n_2) = x(n_1, n_2) + 2x(n_1 - 1, n_2) \\ + 0.4y(n_1 - 1, n_2) + 0.5y(n_1, n_2 - 1),$$

with $\text{supp}\{h\} = \{n_1 \geq 0, n_2 \geq 0\}$. What is the resulting ROC_{*h*}?

6. Consider the 2-D impulse response

$$h(n_1, n_2) = 5\rho_1^{n_1+n_2}u(n_1, n_2) * \rho_2^{n_1-n_2}u(n_1, n_2),$$

where u is the first quadrant unit step function and the ρ_i are real numbers that satisfy $-1 < \rho_i < +1$ for $i = 1, 2$. (Note the convolution indicated by $*$.)

- (a) Find the Z-transform $H(z_1, z_2)$ along with its ROC.
 (b) Can h be the impulse response of a stable filter?
7. This problem concerns stability and recursive filters.
 (a) For what values of the parameters a and b is the impulse response $h(n_1, n_2) = a^{|n_1|}b^{|n_2|}$ stable in the bounded-input bounded-output sense?
 (b) For stable values of a and b , find a realization of the filter with this impulse response in terms of four stable 1-D recursive filters running in various directions on the discrete plane.

8. Find the inverse Z-transform of

$$X(z_1, z_2) = \frac{1}{1 - .9z_1^{-1}} \frac{1}{1 - .9z_2^{-1}} + \frac{.9z_1}{1 - .9z_1} \frac{.9z_2}{1 - .9z_2},$$

with $\text{ROC}_x \supset \{|z_1| = |z_2| = 1\}$. Is your resulting x absolutely summable?

9. Prove the Z-transform relationship in (3.4-1) for linear mapping of variables.

10. Find the inverse Z-transform of

$$X(z_1, z_2) = \frac{1}{1 - (z_1^{-1} + z_2^{-1})},$$

with $\text{ROC} = \{(z_1, z_2) \mid |z_1|^{-1} + |z_2|^{-1} < 1\}$ via series expansion.

11. Find the inverse Z-transform of

$$X(z_1, z_2) = \frac{z_1 + z_2}{1 - \frac{1}{2}(z_1^{-1} + z_2^{-1})}, \text{ with } \text{ROC}_x \supset \{|z_1| > 1, |z_2| > 1\}.$$

12. The *multinomial theorem* can be stated as

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{\substack{l_1, l_2, \dots, l_m \\ l_i \geq 0 \text{ and } \sum_{i=1}^m l_i = n}} \binom{n}{l_1, l_2, \dots, l_m} x_1^{l_1} x_2^{l_2} \cdots x_m^{l_m},$$

where n is a positive integer and the $x_i, i = 1$ to m are variables. Here the *multinomial coefficient* is given as

$$\binom{n}{l_1, l_2, \dots, l_m} \triangleq \frac{n!}{l_1! l_2! \cdots l_m!}$$

for non-negative integers l_i that sum to n . Note that this multinomial coefficient is simply related to the binomial coefficient when $m = 2$. In fact, denoting the binomial coefficient as $\binom{n}{l}_B$, we have

$$\binom{n}{l, n-l} = \binom{n}{l}_B.$$

Use the multinomial theorem to find a closed-form expression for the inverse 2-D Z-transform of

$$X(z_1, z_2) = \frac{1}{1 - (az_1^{-1} + bz_2^{-1} + cz_1^{-1}z_2^{-1})}$$

with support of $x(n_1, n_2)$ on the first quadrant—i.e., $\text{supp}(x) = \{n_1 \geq 0, n_2 \geq 0\}$. Your answer should be a finite sum over multinomial coefficients and the variables a, b , and c raised to various integer powers, for each coordinate (n_1, n_2) on the first quadrant. (Hint: First make use of the series expansion

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots$$

for appropriate choice of x .)

13. Show that an $N \times N$ support signal can always be written as the sum of N or fewer separable factors by regarding the signal as a matrix and applying the singular value decomposition [11].
14. Use Z-transform root maps to numerically test the following filter for stability in the ++ causal sense:

$$H(z_1, z_2) = \frac{1}{1 - \frac{1}{2}z_1^{-1} - \frac{1}{4}z_1^{-1}z_2^{-1} - \frac{1}{4}z_2^{-2}}.$$

Use MATLAB and the functions `ROOTS` and `POLY`.

15. Prove Theorem 3.5–3, the Z-transform stability condition for a spatial filter $H(z_1, z_2) = 1/A(z_1, z_2)$ with third quadrant support impulse response $h(n_1, n_2)$.
16. Consider the following 2-D difference equation:

$$y(n_1, n_2) + 2y(n_1 - 1, n_2) + 3y(n_1, n_2 - 1) + 7y(n_1 - 1, n_2 - 1) = x(n_1, n_2).$$

In answering the following questions, use the standard image-processing coordinate system: n_1 axis horizontal and n_2 axis downward.

- (a) Find a *stable* direction of recursion for this equation.
- (b) Sketch the impulse response support of the resulting system of part (a).

- (c) Find the Z-transform system function along with its associated ROC for the resulting system of part (a).

17. Consider the 3-D linear filter

$$H(z_1, z_2, z_3) = \frac{B(z_1, z_2, z_3)}{A(z_1, z_2, z_3)}, \text{ where } a(0, 0, 0) = 1,$$

where z_1 and z_2 correspond to spatial dimensions and z_3 corresponds to the time dimension. Assume that the impulse response $h(n_1, n_2, n_3)$ has *first octant* support in the following parts (a) and (b). (Note: Ignore any contribution of the numerator to system stability.)

- (a) If this first octant filter is stable, show that the ROC of $H(z_1, z_2, z_3)$ must include $\{|z_1| = |z_2| = |z_3| = 1\}$. Then extend this ROC as appropriate for the specified first octant support of the impulse response $h(n_1, n_2, n_3)$. Hence, conclude that *stability implies that* $A(z_1, z_2, z_3)$ cannot equal zero in this region; that is, this is a *necessary* condition for stability for first octant filters. What is this stability region?
- (b) Show that the condition that $A(z_1, z_2, z_3) \neq 0$ on the stability region of part (a) is a *sufficient* condition for stability of first octant filters.
- (c) Let's say that a 3-D filter is *causal* if its impulse response has support on $\{\mathbf{n} | n_3 \geq 0\}$. Note that there is now no restriction on the spatial support, (i.e., in the n_1 and n_2 dimensions). What is the stability region of a *causal* 3-D filter?
18. Consider computing the output of a NSHP filter on a quarter-plane region. Specifically, the filter is

$$y(n_1, n_2) = 0.2y(n_1 - 1, n_2) + 0.4y(n_1, n_2 - 1) + 0.3y(n_1 + 1, n_2 - 1) + x(n_1, n_2),$$

and the solution region is $\{n_1 \geq 0, n_2 \geq 0\}$, with a boundary condition of zero along the two edges $n_1 = -1$ and $n_2 = -1$. This then specifies a system T mapping quarter-plane inputs x into quarter-plane outputs y (i.e., $y = T[x]$).

- (a) Is T a linear operator?
- (b) Is T a shift-invariant operator?
- (c) Is T a stable operator?

REFERENCES

- [1] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [2] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [3] N. Levinson and R. M. Redheffer, *Complex Variables*, Holden-Day, San Francisco, 1970.
- [4] S. G. Krantz, *Function Theory of Several Complex Variables*, Wiley-Interscience, New York, 1982.

- [5] D. M. Goodman, "Some Stability Properties of Two-Dimensional Linear Shift-Invariant Digital Filters," *IEEE Trans. on Video Tech.*, vol. CAS-24, pp. 201–209, April 1977.
- [6] G. A. Bliss, *Algebraic Functions*, Dover, NY, 1966.
- [7] J. L. Shanks, S. Treitel, and J. H. Justice, "Stability and Synthesis of Two-Dimensional Recursive Filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 115–128, June 1972.
- [8] T. S. Huang, "Stability of Two-Dimensional Recursive Filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 158–163, June 1972.
- [9] M. G. Strintzis, "Test of Stability of Multidimensional Filters," *IEEE Trans. on Video Tech.*, vol. CAS-24, pp. 432–437, August 1977.
- [10] M. P. Ekstrom and J. W. Woods, "Two-Dimensional Spectral Factorization with Applications in Recursive Digital Filtering," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 115–128, April 1976.
- [11] G. Strang, *Linear Algebra and its Applications*, Academic Press, New York, 1976.

2-D Discrete-Space Transforms

In this chapter we look at discrete-space transforms such as discrete Fourier series, discrete Fourier transform (DFT), and discrete cosine transform (DCT) in two dimensions. We also discuss fast and efficient realizations of the DFT and DCT. The DFT is a heavily used tool in image and multidimensional signal processing. Block transforms can be obtained from scanning the data into small blocks and then performing the DFT or DCT on each block. The block DCT is used extensively in image and video compression for transmission and storage. We also consider the subband/wavelet transform (SWT), which can be considered as a generalization of the block DCT transform wherein the basis functions are allowed to overlap from block to block. These SWTs can also be considered as a generalization of the Fourier transform wherein resolution in space can be traded off versus resolution in frequency.

4.1 DISCRETE FOURIER SERIES

The discrete Fourier series (DFS) has been called “the Fourier transform for periodic sequences,” in that it plays the same role for them that the Fourier transform plays for nonperiodic (ordinary) sequences. The DFS also provides a theoretical stepping stone toward the DFT, which has great practical significance in signal and image processing as a Fourier transform for finite support sequences. Actually, we can take the Fourier transform of periodic sequences, but only with impulse functions. The DFS can give an equivalent representation without the need for Dirac impulses.

Since this section will mainly be concerned with periodic sequences, we establish the following convenient notation.

Notation: We write $\tilde{x}(n_1, n_2)$ to denote a sequence that is rectangularly periodic with period $N_1 \times N_2$ when only one period is considered. When two or more periods are involved in a problem, we will extend this notation and denote the periods explicitly.

Definition 4.1–1: Discrete Fourier Series

For a periodic sequence $\tilde{x}(n_1, n_2)$, with rectangular period $N_1 \times N_2$ for positive integers N_1 and N_2 , we define its DFS transform as

$$\tilde{X}(k_1, k_2) \triangleq \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}(n_1, n_2) \exp -j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right),$$

for all integers k_1 and k_2 in the lattice Z^2 (i.e., $-\infty < k_1, k_2 < +\infty$).¹

Since k_1 and k_2 are integers, we can easily see that the DFS is itself periodic with period $N_1 \times N_2$; thus the DFS transforms or maps periodic sequences into periodic sequences. It may be interesting to note that the Fourier transform for discrete space does not share the analogous property, since it maps discrete-space functions into continuous-space functions over $[-\pi, +\pi]^2$. Even though the Fourier transform over continuous space did map this space into itself, its discrete-space counterpart did not. In part because of this self-mapping feature of the DFS, the inverse DFS formula is very similar to it.

Theorem 4.1–1: Inverse DFS

Given $\tilde{X}(k_1, k_2)$, the DFS of periodic sequence $\tilde{x}(n_1, n_2)$, the inverse DFS (or IDFS) is given as

$$\tilde{x}(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \tilde{X}(k_1, k_2) \exp +j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right),$$

for all $-\infty < n_1, n_2 < +\infty$.

Proof We start by inserting the DFS into this claimed inversion formula:

$$\begin{aligned} & \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \left[\sum_{n'_1=0}^{N_1-1} \sum_{n'_2=0}^{N_2-1} \tilde{x}(n'_1, n'_2) \exp -j2\pi \left(\frac{n'_1 k_1}{N_1} + \frac{n'_2 k_2}{N_2} \right) \right] \\ & \quad \times \exp +j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \\ &= \sum_{n'_1=0}^{N_1-1} \sum_{n'_2=0}^{N_2-1} \tilde{x}(n'_1, n'_2) \left\{ \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \exp +j2\pi \left[\frac{(n_1 - n'_1)k_1}{N_1} + \frac{(n_2 - n'_2)k_2}{N_2} \right] \right\} \\ &= \sum_{n'_1=0}^{N_1-1} \sum_{n'_2=0}^{N_2-1} \tilde{x}(n'_1, n'_2) \left[\sum_{k_1=-\infty}^{+\infty} \delta(n_1 - n'_1 + k_1 N_1) \sum_{k_2=-\infty}^{+\infty} \delta(n_2 - n'_2 + k_2 N_2) \right] \\ &= \tilde{x}((n_1)_{N_1}, (n_2)_{N_2}), \end{aligned}$$

¹Note that $-\infty < k_1, k_2 < +\infty$ is shorthand for $-\infty < k_1 < +\infty$ and $-\infty < k_2 < +\infty$.

i.e., a periodic extension of \tilde{x} off its base period $[0, N_1 - 1] \times [0, N_2 - 1]$.

$$= \tilde{x}(n_1, n_2),$$

since \tilde{x} is periodic with rectangular period $N_1 \times N_2$, thus completing the proof. ■

In fact, one could just take the DFS of a DFS. The reader may care to show that the result would be a scaled and reflected-through-the-origin version of the original periodic function \tilde{x} . The following example calculates and plots the DFS of a simple periodic discrete-space pulse function.

Example 4.1–1: Calculation of a DFS

Let $\tilde{x}(n_1, n_2) = I_{4 \times 4}(n_1, n_2)$ for $0 \leq n_1, n_2 \leq 7$, where I_S is the *indicator function*² of the coordinate set S . Here, N_1 and $N_2 = 8$. Looking at the resulting periodic function, we see a periodic pulse function on the plane with 25% ones and 75% zeros, which could serve as an optical image test pattern of sorts. Then

$$\begin{aligned} \tilde{X}(k_1, k_2) &= \sum_{n_1=0}^3 \sum_{n_2=0}^3 W_8^{n_1 k_1} W_8^{n_2 k_2}, \text{ with } W_N \triangleq \exp -j \frac{2\pi}{N} \\ &= \sum_{n_1=0}^3 W_8^{n_1 k_1} \sum_{n_2=0}^3 W_8^{n_2 k_2} \\ &= \frac{1 - W_8^{4k_1}}{1 - W_8^{k_1}} \frac{1 - W_8^{4k_2}}{1 - W_8^{k_2}} \\ &= W_8^{\frac{3}{2}k_1} W_8^{\frac{3}{2}k_2} \frac{\sin\left(\frac{\pi k_1}{2}\right)}{\sin\left(\frac{\pi k_1}{8}\right)} \frac{\sin\left(\frac{\pi k_2}{2}\right)}{\sin\left(\frac{\pi k_2}{8}\right)}. \end{aligned}$$

We give a perspective plot of the amplitude part

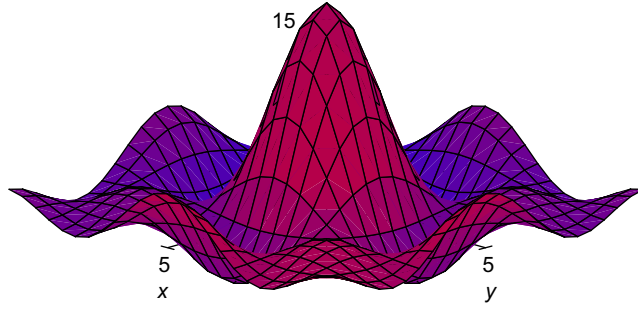
$$\frac{\sin\left(\frac{\pi k_1}{2}\right) \sin\left(\frac{\pi k_2}{2}\right)}{\sin\left(\frac{\pi k_1}{8}\right) \sin\left(\frac{\pi k_2}{8}\right)},$$

in Figure 4.1–1. ■

Properties of the DFS Transform

The properties of the DFS are similar to those of the Fourier transform but distinct in several important aspects, arising both from the periodic nature of \tilde{x} as well as

²The indicator function I_S gives a 1 for a coordinate (n_1, n_2) contained in the set S and a 0 elsewhere, thereby *indicating* the set S .

**FIGURE 4.1–1**

Plot of amplitude part of the DFS in [Example 4.1–1](#).

from the fact that the frequency arguments k_1 and k_2 are integers. We consider two sequences \tilde{x} and \tilde{y} with the same rectangular period $N_1 \times N_2$, having DFS transforms \tilde{X} and \tilde{Y} , respectively. We offer some proofs below.

1. *Linearity:*

$$a\tilde{x} + b\tilde{y} \Leftrightarrow a\tilde{X} + b\tilde{Y},$$

when both sequences have the same period $N_1 \times N_2$.

2. *Periodic convolution:*

We define periodic convolution with the operator symbol \otimes , for two periodic sequences with the same period as:

$$(\tilde{x} \otimes \tilde{y})(n_1, n_2) \triangleq \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} \tilde{x}(l_1, l_2) \tilde{y}(n_1 - l_1, n_2 - l_2).$$

We then have the following transform pair:

$$(\tilde{x} \otimes \tilde{y})(n_1, n_2) \Leftrightarrow \tilde{X}(k_1, k_2) \tilde{Y}(k_1, k_2).$$

The proof is given below.

3. *Multiplication:*

$$\tilde{x}(n_1, n_2) \tilde{y}(n_1, n_2) \Leftrightarrow \frac{1}{N_1 N_2} \tilde{X}(k_1, k_2) \otimes \tilde{Y}(k_1, k_2).$$

4. *Separability:*

$$\tilde{x}_1(n_1) \tilde{x}_2(n_2) \Leftrightarrow \tilde{X}_1(k_1) \tilde{X}_2(k_2),$$

the product of a 1-D N_1 -point and a 1-D N_2 -point DFS, respectively.

5. *Shifting (delay):*

$$\tilde{x}(n_1 - m_1, n_2 - m_2) \Leftrightarrow \tilde{X}(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right),$$

where the shift vector (m_1, m_2) is integer valued. The proof is given below.

6. Parseval's theorem:

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}(n_1, n_2) \tilde{y}^*(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \tilde{X}(k_1, k_2) \tilde{Y}^*(k_1, k_2),$$

with a special case for $x = y$, the “energy balance formula,” since the left-hand side can be viewed as the “energy” in one period,

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} |\tilde{x}(n_1, n_2)|^2 = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} |\tilde{X}(k_1, k_2)|^2.$$

The proof is assigned as an end-of-chapter problem.

7. Symmetry properties:

(a) Conjugation:

$$\tilde{x}^*(n_1, n_2) \Leftrightarrow \tilde{X}^*(-k_1, -k_2).$$

(b) Arguments reversed (reflection through origin):

$$\tilde{x}(-n_1, -n_2) \Leftrightarrow \tilde{X}(-k_1, -k_2).$$

(c) Real-valued sequences (special case): By the conjugation property (a), applying it to a real-valued sequence \tilde{x} , we have the *conjugate symmetry* property,

$$\tilde{X}(k_1, k_2) = \tilde{X}^*(-k_1, -k_2).$$

From this equation, the following four important symmetry properties of real periodic bi-sequences follow easily:

i. $\text{Re} \tilde{X}(k_1, k_2)$ is even through the origin:

$$\text{Re} \tilde{X}(k_1, k_2) = \text{Re} \tilde{X}(-k_1, -k_2).$$

ii. $\text{Im} \tilde{X}(k_1, k_2)$ is odd through the origin:

$$\text{Im} \tilde{X}(k_1, k_2) = -\text{Im} \tilde{X}(-k_1, -k_2).$$

iii. $|\tilde{X}(k_1, k_2)|$ is even through the origin:

$$|\tilde{X}(k_1, k_2)| = |\tilde{X}(-k_1, -k_2)|.$$

iv. $\arg \tilde{X}(k_1, k_2)$ is odd through the origin:

$$\arg \tilde{X}(k_1, k_2) = -\arg \tilde{X}(-k_1, -k_2).$$

These last properties are used for reducing required data storage for the DFS by an approximate factor of 1/2 in the real-valued image case.

Periodic Convolution

In 2-D, periodic convolution is very similar to the 1-D case for periodic sequences $\tilde{x}(n)$ of one variable. Rectangular periodicity, as considered here, allows an easy generalization. As in the 1-D case, we note that regular convolution is not possible with periodic sequences, since they have infinite energy.

We offer next a proof of the periodic convolution property.

Proof of property 2: Let

$$\begin{aligned}\tilde{y}(n_1, n_2) &\triangleq (\tilde{x}_1 \otimes \tilde{x}_2)(n_1, n_2) \\ &= \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} \tilde{x}_1(l_1, l_2) \tilde{x}_2(n_1 - l_1, n_2 - l_2).\end{aligned}$$

Then

$$\begin{aligned}\tilde{Y}(k_1, k_2) &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{y}(n_1, n_2) \exp -j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \\ &= \sum_{n_1, n_2} \sum_{l_1, l_2} \tilde{x}_1(l_1, l_2) \tilde{x}_2(n_1 - l_1, n_2 - l_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2},\end{aligned}$$

with $W_N \triangleq \exp -j\frac{2\pi}{N}$. Thus

$$\begin{aligned}\tilde{Y}(k_1, k_2) &= \sum_{n_1, n_2} \sum_{l_1, l_2} \tilde{x}_1(l_1, l_2) W_{N_1}^{l_1 k_1} W_{N_2}^{l_2 k_2} \left[\tilde{x}_2(n_1 - l_1, n_2 - l_2) W_{N_1}^{(n_1 - l_1) k_1} W_{N_2}^{(n_2 - l_2) k_2} \right] \\ &= \sum_{l_1, l_2} \tilde{x}_1(l_1, l_2) W_{N_1}^{l_1 k_1} W_{N_2}^{l_2 k_2} \left[\sum_{n_1, n_2} \tilde{x}_2(n_1 - l_1, n_2 - l_2) W_{N_1}^{(n_1 - l_1) k_1} W_{N_2}^{(n_2 - l_2) k_2} \right] \\ &= \sum_{l_1, l_2} \tilde{x}_1(l_1, l_2) W_{N_1}^{l_1 k_1} W_{N_2}^{l_2 k_2} [\tilde{X}_2(k_1, k_2)],\end{aligned}$$

by periodicity,

$$= \tilde{X}_1(k_1, k_2) \tilde{X}_2(k_1, k_2),$$

as was to be shown.

Shifting or Delay Property

According to DFS property 5 in this section, Properties of the DFS Transform, $\tilde{x}(n_1 - m_1, n_2 - m_2)$ has DFS $\tilde{X}(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right)$, which can be seen directly as follows. First, by definition,

$$\tilde{X}(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}(n_1, n_2) \exp -j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right),$$

so the DFS of $\tilde{x}(n_1 - m_1, n_2 - m_2)$ is given by

$$\begin{aligned}&\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}(n_1 - m_1, n_2 - m_2) \exp -j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \\ &= \sum_{n'_1=-m_1}^{N_1-1-m_1} \sum_{n'_2=-m_2}^{N_2-1-m_2} \tilde{x}(n'_1, n'_2) \exp -j2\pi \left[\frac{(n'_1 + m_1) k_1}{N_1} + \frac{(n'_2 + m_2) k_2}{N_2} \right]\end{aligned}$$

$$\begin{aligned}
&= \left[\sum_{n'_1=-m_1}^{N_1-1-m_1} \sum_{n'_2=-m_2}^{N_2-1-m_2} \tilde{x}(n'_1, n'_2) \exp -j2\pi \left(\frac{n'_1 k_1}{N_1} + \frac{n'_2 k_2}{N_2} \right) \right] \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right), \\
&= \left[\sum_{n'_1=0}^{N_1-1} \sum_{n'_2=0}^{N_2-1} \tilde{x}(n'_1, n'_2) \exp -j2\pi \left(\frac{n'_1 k_1}{N_1} + \frac{n'_2 k_2}{N_2} \right) \right] \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right) \\
&= \tilde{X}(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right),
\end{aligned}$$

where we have substituted $n'_1 \triangleq n_1 - m_1$ and $n'_2 \triangleq n_2 - m_2$ in the first step, and where the second to last line follows from the rectangular periodicity of \tilde{x} .

Comments

1. If two rectangularly periodic sequences have different periods, we can use the DFS properties if we first find their common period, given as $N_i = \text{LCM}(N_i^x, N_i^y)$, where sequence \tilde{x} is periodic $N_1^x \times N_2^x$ and similarly for \tilde{y} . (Here, LCM stands for *least common multiple*.)
2. The DFS separability property 4 in Section 4.1 comes about because the DFS operator is a separable operator and the operand $\tilde{x}(n_1, n_2)$ is assumed there to be a separable function. A problem at the end of the chapter explores whether this is an exception to the rule that “multiplication in the space domain corresponds to convolution in the frequency domain,” or not.

4.2 DISCRETE FOURIER TRANSFORM

The discrete Fourier transform (DFT) is “the Fourier transform for finite length sequences” because, unlike the Fourier transform, the DFT has a discrete argument and can be stored in a finite number of infinite word-length locations. Yet, still it turns out that the DFT can be used to exactly implement convolution for finite size arrays. Following [1, 2], our approach to the DFT will be through the DFS, which is made possible by the isomorphism between rectangular periodic and finite length rectangular support sequences.

Definition 4.2–1: Discrete Fourier Transform

For a finite support sequence $x(n_1, n_2)$ with support $[0, N_1 - 1] \times [0, N_2 - 1]$, we define its DFT $X(k_1, k_2)$ for integers k_1 and k_2 as follows:

$$X(k_1, k_2) \triangleq \begin{cases} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2}, & (k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1] \\ 0, & \text{else.} \end{cases} \quad (4.2-1)$$

We note that the DFT maps finite support rectangular sequences into themselves. Images of the DFT basis functions of size 8×8 are shown next, with real parts in Figure 4.2-1 and imaginary parts in Figure 4.2-2. The real parts of the basis functions represent the components of x that are symmetric with respect to the 8×8 square, while the imaginary parts of these basis functions represent the nonsymmetric parts. In these figures, the color *white* is maximum positive (+1), *midgray* is 0, and *black* is minimum negative (-1). Each basis function occupies a small square, and the squares are then arranged into an 8×8 mosaic. Note that the highest frequencies are in the middle at $(k_1, k_2) = (4, 4)$ and correspond to the Fourier transform

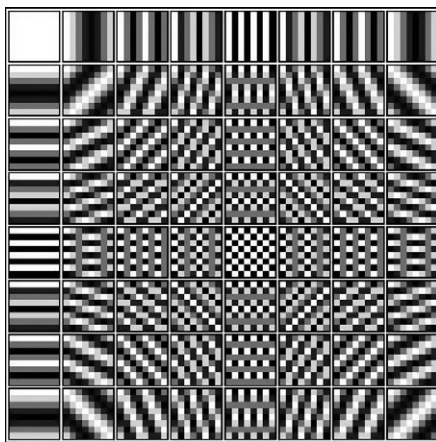


FIGURE 4.2-1

Image of the real part of 8×8 DFT basis functions.

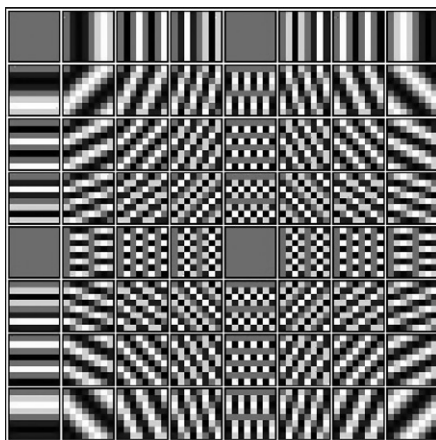


FIGURE 4.2-2

Image of the imaginary part of 8×8 DFT basis functions.

at $(\omega_1, \omega_2) = (\pi, \pi)$. So the DFT is seen as a projection of the finite support input sequence $x(n_1, n_2)$ onto these basis functions. The DFT coefficients then are the representation coefficients for this basis. The inverse DFT (IDFT) exists and is given by

$$x(n_1, n_2) = \begin{cases} \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2}, & (n_1, n_2) \in [0, N_1 - 1] \times [0, N_2 - 1], \\ 0, & \text{else.} \end{cases} \quad (4.2-2)$$

This can be seen as a representation of x in terms of the basis functions $\frac{1}{N_1 N_2} W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2}$ and the expansion coefficients $X(k_1, k_2)$.

The correctness of this 2-D IDFT formula can be seen in a number of different ways. Perhaps the easiest at this point is to realize that the 2-D DFT is a separable transform; as such, we can realize it as the concatenation of two 1-D DFTs. Since we can see that the 2-D IDFT is just the inverse of each of these 1-D DFTs in a given order, say first by row and then by column, we have the desired result based on the known validity of the 1-D DFT/IDFT transform pair, applied twice.

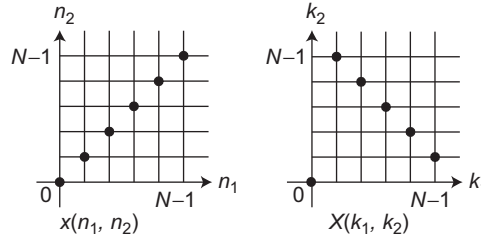
A second method is to rely on the DFS, which we have established in the previous section. Indeed, the main value of the DFS is its use as an aid in understanding DFT properties. The key concept is that rectangular periodic and rectangular finite support sequences are isomorphic to one another; that is, given \tilde{x} , we can define a finite support x as $x \triangleq \tilde{x} I_{N_1 \times N_2}$, and given a finite support x , we can find the corresponding \tilde{x} as $\tilde{x} = x[(n_1)_{N_1}, (n_2)_{N_2}]$, where we use the notation $(n)_N$ meaning “ $n \bmod N$.” Still a third method is to simply insert (4.2-1) into (4.2-2) and perform the 2-D proof directly.

Example 4.2-1: Ones on Diagonal of Square

Consider a spatial sequence of finite extent $x(n_1, n_2) = \delta(n_1 - n_2) I_{N \times N}$, as illustrated in the left-hand side of Figure 4.2-3. Proceeding to take the DFT, we obtain for $(k_1, k_2) \in [0, N - 1] \times [0, N - 1]$ the following:

$$\begin{aligned} X(k_1, k_2) &= \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \delta(n_1 - n_2) W_N^{n_1 k_1} W_N^{n_2 k_2} \\ &= \sum_{n_1=0}^{N-1} W_N^{n_1 (k_1 + k_2)} \\ &= N \delta(k_1 + k_2) \quad \text{for } (k_1, k_2) \in [0, N - 1] \times [0, N - 1] \end{aligned}$$

and, of course, $X(k_1, k_2) = 0$ elsewhere, as illustrated on the right-hand side of Figure 4.2-3. Note that this is analogous to the case of the continuous-parameter Fourier transform of an impulse line (see Chapter 1). This ideal result only occurs when the line is exactly at 0, 45, or 90, or 135 degrees and the DFT size is square. For other angles, we get a $\sin Nk/N \sin k$ type of approximation that tends toward the impulse line solution more and more as N grows larger without bound. Of course, as was true for the continuous-space

**FIGURE 4.2-3**

DFT of ones on diagonal of square.

(parameter) Fourier transform, the line of impulses in frequency is perpendicular, or at 90 degrees, to the impulse line in space. ■

DFT Properties

The properties of the DFT are similar to those of the DFS. The key difference is that the support of the sequence x and of the DFT X is finite. We consider two sequences x and y with the same rectangular $N_1 \times N_2$ support, having DFT transforms X and Y , respectively. We then offer proofs of some of these properties below.

1. *Linearity:*

$$ax + by \Leftrightarrow aX + bY,$$

when both sequences have the same support $[0, N_1 - 1] \times [0, N_2 - 1]$.

2. *Periodic convolution:*

We define periodic convolution for two finite support sequences with the same period as

$$(x \otimes y)(n_1, n_2) \triangleq \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} x(l_1, l_2) y[(n_1 - l_1)_{N_1}, (n_2 - l_2)_{N_2}] I_{N_1 \times N_2}(n_1, n_2), \quad (4.2-3)$$

using the operator symbol \otimes . We then have the following transform pair:

$$(x \otimes y)(n_1, n_2) \Leftrightarrow X(k_1, k_2) Y(k_1, k_2).$$

The proof is given in the following section.

3. *Multiplication:*

$$x(n_1, n_2) y(n_1, n_2) \Leftrightarrow \frac{1}{N_1 N_2} X(k_1, k_2) \otimes Y(k_1, k_2).$$

4. *Separability:*

$$x_1(n_1) x_2(n_2) \Leftrightarrow X_1(k_1) X_2(k_2),$$

the product of a 1-D N_1 -point and a 1-D N_2 -point DFT, respectively.

5. *Shifting (delay):*

$$x[(n_1 - m_1)_{N_1}, (n_2 - m_2)_{N_2}] I_{N_1 \times N_2}(n_1, n_2) \\ \Leftrightarrow X(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right),$$

where the shift vector (m_1, m_2) is integer valued. The proof is given in the following section.

6. *Parseval's theorem:*

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) y^*(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) Y^*(k_1, k_2),$$

with special case for $x = y$, the “energy balance formula,” since the left-hand side then becomes the energy of the signal

$$\mathcal{E}_x = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} |x(n_1, n_2)|^2 = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} |X(k_1, k_2)|^2.$$

The proof is assigned as end-of-chapter problem 8.

7. *Symmetry properties:*

(a) *Conjugation:*

$$x^*(n_1, n_2) \Leftrightarrow X^*[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2).$$

(b) *Arguments reversed (modulo reflection through origin):*

$$x[(-n_1)_{N_1}, (-n_2)_{N_2}] I_{N_1 \times N_2}(n_1, n_2) \Leftrightarrow X[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2).$$

(c) *Real-valued sequences (a special case):* By applying the preceding conjugation property to a real-valued sequence \tilde{x} , we have the *conjugate symmetry* property

$$X(k_1, k_2) = X^*[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2). \quad (4.2-4)$$

From this equation, the following four properties follow easily:

i. $\text{Re}X(k_1, k_2)$ is *even*:

$$\text{Re}X(k_1, k_2) = \text{Re}X[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2).$$

ii. $\text{Im}X(k_1, k_2)$ is *odd*:

$$\text{Im}X(k_1, k_2) = -\text{Im}X[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2).$$

iii. $|X(k_1, k_2)|$ is *even*:

$$|X(k_1, k_2)| = |X[(-k_1)_{N_1}, (-k_2)_{N_2}] I_{N_1 \times N_2}(k_1, k_2)|.$$

iv. $\arg X(k_1, k_2)$ may be taken as³ *odd*:

$$\arg X(k_1, k_2) = -\arg X(k_1, k_2).$$

These last properties are used for reducing required data storage for the DFT by an approximate factor of 1/2 in the real-valued image case.

Proof of DFT Circular Convolution Property 2

Key property 2 says that multiplication of 2-D DFTs corresponds to the defined circular convolution in the spatial domain, in a manner very similar to that in one dimension. In fact, we can see from (4.2–3) that this operation is separable into the 1-D circular convolution along the rows, followed by the 1-D circular convolution over the columns. The correctness of this property can then be proved by using the 1-D proof twice; once for the rows and once for the columns. Another way to see the result is to consider the corresponding periodic sequences \tilde{x} and \tilde{y} . Then (4.2–3) can be seen as the first period from their periodic convolution since $\tilde{x}(n_1, n_2) = x[(n_1)_{N_1}, (n_2)_{N_2}]$ and $\tilde{y} = y[(n_1)_{N_1}, (n_2)_{N_2}]$. The first period of their DFS product must be the corresponding DFT result; thus, by property 2 of DFS, we have the result here, since by the correspondence $\tilde{x} \sim x$, it must be that $\tilde{X} \sim X$.

Proof of DFT Circular Shift Property 5

Since $\tilde{x}(n_1, n_2) = x[(n_1)_{N_1}, (n_2)_{N_2}]$, it follows that the periodic shift of \tilde{x} agrees with the circular shift of x ,

$$\tilde{x}(n_1 - m_1, n_2 - m_2) = x[(n_1 - m_1)_{N_1}, (n_2 - m_2)_{N_2}],$$

for $(n_1, n_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$, so the DFS of the left-hand side must equal the DFT of the right-hand side, over the fundamental period in frequency; that is, $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. We thus have the DFT

$$\begin{aligned} \tilde{X}(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right) &= I_{N_1 \times N_2}(k_1, k_2) \\ &= X(k_1, k_2) \exp -j2\pi \left(\frac{m_1 k_1}{N_1} + \frac{m_2 k_2}{N_2} \right). \end{aligned}$$

An end-of-chapter problem asks you to use this circular shift property to prove the 2-D DFT circular convolution property directly in the 2-D DFT domain.

Example 4.2–2: 2-D Circular Convolution

Let $N_1 = N_2 = 4$. The diagram in Figure 4.2–4 shows an example of the 2-D circular convolution of two small arrays x and y . In this figure, the two top plots show the arrays $x(n_1, n_2)$ and $y(n_1, n_2)$, where the open circles indicate zero values of these 4×4 support signals. The nonzero values are denoted by filled-in circles in the 3×3 triangle support x , and various filled-in shapes on the square 2×2 support y . To perform their convolution,

³We cannot use “is” here because you have to select the right multiple of 2π to add to the phase curve to make it “odd,” since phase is a multiple-valued function.

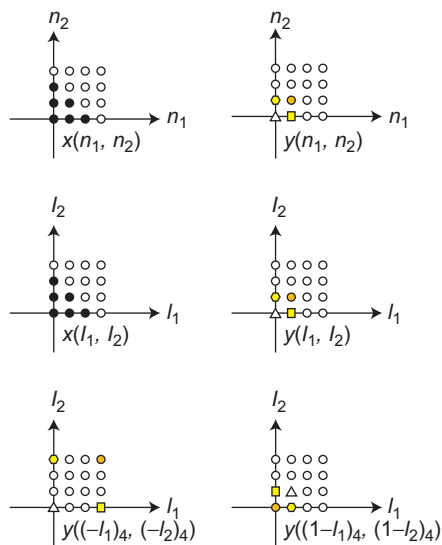


FIGURE 4.2-4

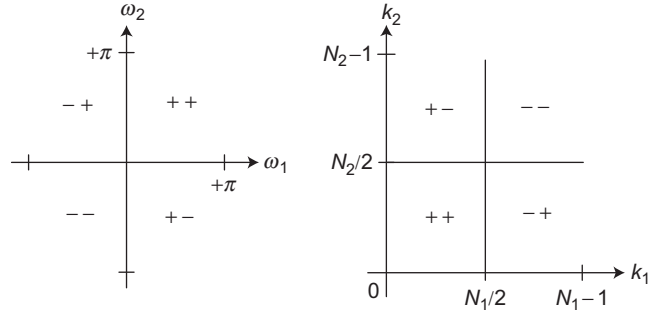
Example of 2-D circular convolution of small triangle support x and small square support y , both considered as $N_1 \times N_2 = 4 \times 4$ support.

over dummy variables (l_1, l_2) , the middle plots of the figure show the inputs as functions of these dummy variables. The bottom line then shows plots of y circularly shifted to correspond to the output points $(n_1, n_2) = (0, 0)$ on the left and $(n_1, n_2) = (1, 1)$ on the right. Continuing in this way, we can see that for all (n_1, n_2) we get the linear convolution result. This has happened because the circular wrap-around points in y only occur at zero values for x . We see that this was due to the larger value we took for N_1 and N_2 , i.e., larger than the necessary value to hold the output. ■

We see in this example that the circular or wrap-around does not affect all output points. In fact, by enclosing the small triangular support signal and the small pulse in a larger 4×4 square, we have avoided the wrap-around for all $(n_1 \geq 1, n_2 \geq 1)$. We can generalize this result as follows. Let the supports of signals x and y be given as

$$\text{supp}\{x\} = [0, M_1 - 1] \times [0, M_2 - 1] \quad \text{and} \quad \text{supp}\{y\} = [0, L_1 - 1] \times [0, L_2 - 1].$$

Then if the DFT size is taken as $N_1 = M_1 + L_1 - 1$ or larger, and $N_2 = M_2 + L_2 - 1$ or larger, we get the linear convolution result. Thus, to avoid the circular or spatial-aliasing result, we simply have to pad the two signals with zeros out to a DFT size that is large enough to contain the *linear convolution* result—the result of ordinary noncircular convolution.

**FIGURE 4.2-5**

Mapping of FT samples to DFT locations.

Relation of DFT to Fourier Transform

For a finite support sequence x , we can compute both the Fourier transform $X(\omega_1, \omega_2)$ as well as the DFT $X(k_1, k_2)$. We now answer the question of the relation of these two. Comparing (1.2-1) of Chapter 1 to (4.2-1), we have

$$X(k_1, k_2) = X(\omega_{k_1}, \omega_{k_2}), \quad \text{where } \omega_{k_i} \triangleq 2\pi k_i / N_i, \quad i = 1, 2, \quad (4.2-5)$$

for $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. This means that the central or primary period of the Fourier transform $X(\omega_1, \omega_2)$ is not what is sampled, but rather an area of equal size in its first quadrant. Of course, by periodicity this is equivalent to sampling in the primary period. Based on such considerations, we can construct the diagram of Figure 4.2-5 indicating where the DFT samples come from in the Fourier transform plane. In particular, all the FT samples along the ω_i axes map into $k_i = 0$ in the DFT, and the possible samples (which only occur for N_i even) at $\omega_i = \pi$ map into $k_i = N_i/2$. Also note the indicated mapping of the four quadrants of $[-\pi, +\pi]^2$ in the $\omega_1 \times \omega_2$ plane onto $[0, N_1 - 1] \times [0, N_2 - 1]$ in the first quadrant of the $k_1 \times k_2$ plane.

Example 4.2-3: DFT Symmetry in Real-valued Signal Case

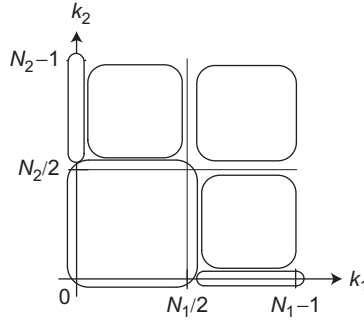
When the image (signal) is real valued, we have the following DFT symmetry property: $X(k_1, k_2) = X^*((-k_1)_{N_1}, (-k_2)_{N_2}) I_{N_1 \times N_2}(k_1, k_2)$. When the DFT is stored, we only need to consider the locations $[0, N_1 - 1] \times [0, N_2 - 1]$. For these locations, we can then write

$$X(k_1, 0) = X^*(N_1 - k_1, 0) \quad \text{for } k_2 = 0,$$

$$X(0, k_2) = X^*(0, N_2 - k_2) \quad \text{for } k_1 = 0,$$

$$X(k_1, k_2) = X^*(N_1 - k_1, N_2 - k_2) \quad \text{otherwise.}$$

This then gives us the conjugate symmetry through the point $(k_1, k_2) = (N_1/2, N_2/2)$, as shown in Figure 4.2-6. The big square with round corners shows an essential part of the DFT coefficients and comprises locations $\{0 \leq k_1 \leq N_1/2, 0 \leq k_2 \leq N_2/2\}$. The other


FIGURE 4.2-6

An illustration of the conjugate symmetry in DFT storage, for the real-valued image case.

essential part is one of the two smaller squares with rounded corners that share a side with this large square. For example, the upper square, which comprises points $\{1 \leq k_1 < N_1/2, N_2/2 < k_2 \leq N_2 - 1\}$. The other smaller square is not needed by the preceding symmetry condition. Neither are the two narrow regions along the axes. Such symmetry can be used to reduce storage by approximately one-half. We only need to store the coefficients for the resulting *nonsymmetric half-space* region. ■

Effect of Sampling in Frequency

Let $x(n_1, n_2)$ be a general signal with Fourier transform $X(\omega_1, \omega_2)$; then we can sample it as in (4.2-5). If we take the IDFT of the function $X(k_1, k_2)I_{N_1 \times N_2}(k_1, k_2)$, we then obtain

$$y(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2} \quad (4.2-6)$$

$$= \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \left[\sum_{\text{all } m_1, m_2} x(m_1, m_2) W_{N_1}^{m_1 k_1} W_{N_2}^{m_2 k_2} \right] W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2} \quad (4.2-7)$$

$$= \sum_{\text{all } m_1, m_2} x(m_1, m_2) \left[\sum_{\text{all } l_1, l_2} \delta(m_1 - n_1 + l_1 N_1, m_2 - n_2 + l_2 N_2) \right] \quad (4.2-8)$$

which displays the *spatial domain aliasing* caused by sampling in frequency. If the original signal x had a finite support $[0, M_1 - 1] \times [0, M_2 - 1]$ and we took samples of its Fourier transform, satisfying $N_1 \geq M_1$ and $N_2 \geq M_2$, then we would have no overlap in (4.2–8), or equivalently

$$x(n_1, n_2) = y(n_1, n_2)I_{N_1 \times N_2}(n_1, n_2),$$

and will avoid the aliased terms from coming into this spatial support region. One interpretation of (4.2–6) is as a numerical approximation to the IFT, whose exact form is not computationally feasible. We thus see that the substitution of an IDFT for the IFT can result in spatial domain aliasing, which however can be controlled by taking the uniformly spaced samples at high enough density to avoid significant (or any) spatial overlap (alias).

Interpolating the DFT

Since the DFT consists of samples of the FT, as in (4.2–5), for an $N_1 \times N_2$ support signal, we can take the inverse, or IDFT, of these samples to express the original signal x in terms of $N_1 N_2$ samples of its Fourier transform $X(\omega_1, \omega_2)$. Then, taking one more FT, we can write $X(\omega_1, \omega_2)$ in terms of its samples:

$$X(\omega_1, \omega_2) = \text{FT}\{\text{IDFT}\{X(\omega_{k_1}, \omega_{k_2})\}\},$$

thus constituting a 2-D sampling theorem for (rectangular) samples in frequency. Actually, performing the calculation, we proceed

$$\begin{aligned} X(\omega_1, \omega_2) &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left[\frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(2\pi k_1/N_1, 2\pi k_2/N_2) W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2} \right] \\ &\quad \times \exp -j(\omega_1 n_1 + \omega_2 n_2) \\ &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(2\pi k_1/N_1, 2\pi k_2/N_2) \\ &\quad \times \left[\frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} W_{N_1}^{-n_1 k_1} W_{N_2}^{-n_2 k_2} \exp -j(\omega_1 n_1 + \omega_2 n_2) \right] \\ &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(2\pi k_1/N_1, 2\pi k_2/N_2) \frac{1}{N_1} \frac{\sin \frac{N_1}{2} \left(\omega_1 - \frac{2\pi k_1}{N_1} \right)}{\sin \frac{1}{2} \left(\omega_1 - \frac{2\pi k_1}{N_1} \right)} \\ &\quad \times \frac{1}{N_2} \frac{\sin \frac{N_2}{2} \left(\omega_2 - \frac{2\pi k_2}{N_2} \right)}{\sin \frac{1}{2} \left(\omega_2 - \frac{2\pi k_2}{N_2} \right)} \exp -j \frac{N_1 - 1}{2} \left(\omega_1 - \frac{2\pi k_1}{N_1} \right) \\ &\quad \times \exp -j \frac{N_2 - 1}{2} \left(\omega_2 - \frac{2\pi k_2}{N_2} \right). \end{aligned}$$

Upon definition of the 1-D interpolation functions

$$\Phi_i(\omega) \triangleq \frac{1}{N_i} \frac{\sin \frac{N_i}{2} \omega}{\sin \frac{1}{2} \omega} \exp -j \frac{N_i - 1}{2} \omega, \quad \text{for } i = 1, 2,$$

we can write

$$X(\omega_1, \omega_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(2\pi k_1/N_1, 2\pi k_2/N_2) \Phi_1\left(\omega_1 - \frac{2\pi k_1}{N_1}\right) \Phi_2\left(\omega_2 - \frac{2\pi k_2}{N_2}\right).$$

This is completely analogous to the 1-D case [2]. As there, we must note that the key is that we have taken a number of samples $N_1 \times N_2$ that is consistent with the spatial support of the signal x . Of course, larger values of the N_i are permissible, but smaller values would allow spatial aliasing—in which case, this 2-D frequency-domain sampling theorem would not be valid.

4.3 2-D DISCRETE COSINE TRANSFORM

One disadvantage of the DFT for some applications is that the transform X is complex valued, even for real data. A related transform, the discrete cosine transform (DCT), does not have this problem. The DCT is a separate transform and not the real part of a Fourier transform. It is widely used in image and video compression applications (e.g., JPEG and MPEG). It is also possible to use DCT for filtering using a slightly different form of convolution called *symmetric convolution* (see end-of-chapter problem 16).

Definition 4.3–1: 2-D DCT

Assume that the data array has finite rectangular support on $[0, N_1 - 1] \times [0, N_2 - 1]$; then the 2-D DCT is given as

$$X_C(k_1, k_2) \triangleq \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 4x(n_1, n_2) \cos \frac{\pi k_1}{2N_1} (2n_1 + 1) \cos \frac{\pi k_2}{2N_2} (2n_2 + 1), \quad (4.3-1)$$

for $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. Otherwise, $X_C(k_1, k_2) \triangleq 0$.

The DCT basis functions for size 8×8 are shown in Figure 4.3–1. The mapping between the mathematical values and the colors (gray levels) is the same as in the DFT figures earlier. Each basis function occupies a small square, and the squares are then arranged into an 8×8 mosaic. Note that unlike the DFT where the highest frequencies occur near $(N_1/2, N_2/2)$, the highest frequencies of the DCT occur at the highest indices $(k_1, k_2) = (7, 7)$.

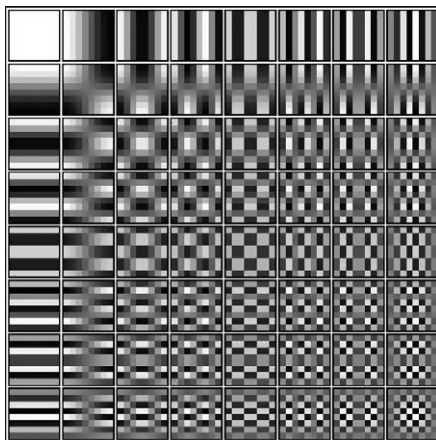
**FIGURE 4.3-1**

Image of the basis functions of the 8×8 DCT with $(0,0)$ in the upper left-hand corner and the k_2 axis pointing downward.

The inverse DCT exists and is given for $(n_1, n_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$ as

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w(k_1) w(k_2) X_C(k_1, k_2) \cos \frac{\pi k_1}{2N_1} (2n_1 + 1) \cos \frac{\pi k_2}{2N_2} (2n_2 + 1),$$

where the weighting function $w(k)$ is given just as in the case of 1-D DCT by

$$w(k) \triangleq \begin{cases} 1/2, & k = 0, \\ 1, & k \neq 0. \end{cases}$$

By observation of (4.3-1), we see that the 2-D DCT is a separable operator. As such it can be applied to the rows and then the columns, or vice versa. Thus the 2-D theory can be developed by repeated application of the 1-D theory. Next, we relate the 1-D DCT to the 1-D DFT of a symmetrically extended sequence. This not only provides an understanding of the DCT but also enables its fast calculation via methods intended for the DFT. Later we present a fast DCT calculation that can avoid the use of complex arithmetic in the usual case where x is a real-valued signal (e.g., an image).

The next two subsections can be skipped by the reader familiar with the 1-D DCT theory.

Review of 1-D DCT

In the 1-D case, the DCT is defined as

$$X_C(k) \triangleq \begin{cases} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi k}{2N} (2n + 1), & k \in [0, N - 1], \\ 0, & \text{else,} \end{cases} \quad (4.3-2)$$

for every N -point signal x having support $[0, N-1]$. The corresponding inverse transform, or IDCT, can be written as

$$x(n) = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} w(k) X_C(k) \cos \frac{\pi k}{2N} (2n+1), & n \in [0, N-1], \\ 0, & \text{else.} \end{cases} \quad (4.3-3)$$

It turns out that this 1-D DCT can be understood in terms of the DFT of a *symmetrically extended sequence*⁴

$$y(n) \triangleq x(n) + x(2N-1-n), \quad (4.3-4)$$

with support $[0, 2N-1]$. In fact, on defining the $2N$ point DFT $Y(k) \triangleq \text{DFT}_{2N}\{y(n)\}$, we will show that the DCT is alternatively expressed as

$$X_C(k) = \begin{cases} W_{2N}^{k/2} Y(k), & k \in [0, N-1], \\ 0, & \text{else.} \end{cases} \quad (4.3-5)$$

Thus the DCT is just the DFT analysis of the symmetrically extended signal (4.3-4). Looking at this equation, we see that there is no overlap in its two components, which fit together without a gap. We can see that right after $x(N-1)$ comes $x(N-1)$ at position $n = N$, which is then followed by the rest of the nonzero part of x in reverse order, out to $n = 2N-1$, where sits $x(0)$. We can see a point of symmetry midway between $n = N-1$ and N (i.e., at $n = N - \frac{1}{2}$). If we consider its periodic extension $\tilde{y}(n)$, we will also see a symmetry about the point $n = -\frac{1}{2}$. We thus expect that the $2N$ -point DFT $Y(k)$ will be real valued except for the phase factor $W_{2N}^{-k/2}$. So the phase factor in (4.3-5) is just what is needed to cancel out the phase term in Y and make the DCT real, as it must if the two equations (4.3-1) and (4.3-5) are to agree for real-valued inputs x .

To reconcile these two definitions, we start with (4.3-5) and proceed as follows:

$$\begin{aligned} Y(k) &= \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{nk} \\ &= \sum_{n=0}^{N-1} x(n) W_{2N}^{nk} + \sum_{n'=0}^{N-1} x(n') W_{2N}^{-(n'+1)k}, \text{ with } n' \triangleq 2N-1-n \\ &= W_{2N}^{-k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{(n+.5)k} + W_{2N}^{-k/2} \sum_{n=0}^{N-1} x(n) W_{2N}^{-(n+.5)k} \\ &= W_{2N}^{-k/2} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi k}{2N} (2n+1), \text{ for } k \in [0, 2N-1], \end{aligned}$$

⁴This is not the only way to symmetrically extend x , but this method results in the most widely used, sometimes called DCT-2 [2].

the last line following from $W_{2N}^{-(n+.5)k} = \exp(j2\pi(n+.5)k/2N)$ and Euler's relation, which agrees with the original definition (4.3-2).

The formula for the inverse DCT (4.3-3) can be established similarly, starting from

$$x(n) = \left[\frac{1}{2N} \sum_{k=0}^{2N-1} Y(k) W_{2N}^{-nk} \right] I_N(n).$$

Example 4.3-1: 1-D DCT

In this example, we use MATLAB to take the DCT of the finite support function x given on its support region $[0, N-1]$ as

$$x(n) = 20(1 - n/N).$$

We choose $N = 16$ and call the MATLAB function `dct` using the .m file `DCTeg2.m` that is found at this book's Web site:

```
clear
N=16;
for il=1:N,
    n=il-1;
    signal(il)=20*(1-n/N);
end
stem(signal)
figure
x=fft(signal);
xm=abs(x);
stem(xm)
figure
xc=dct(signal);
stem(xc),
```

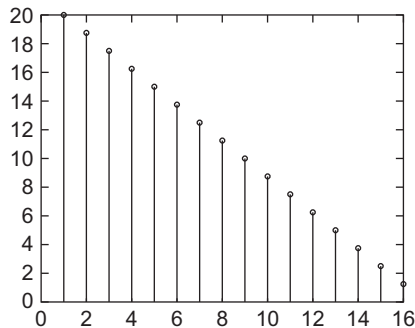


FIGURE 4.3-2

MATLAB plot of $x(n)$ over its support.

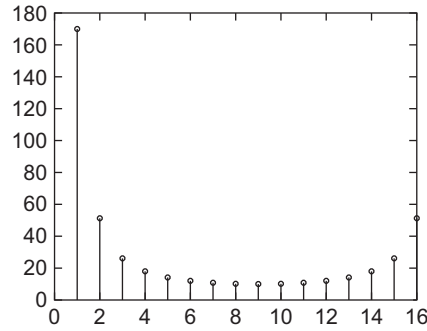


FIGURE 4.3-3

MATLAB plot of DFT magnitude $|X(k)|$.

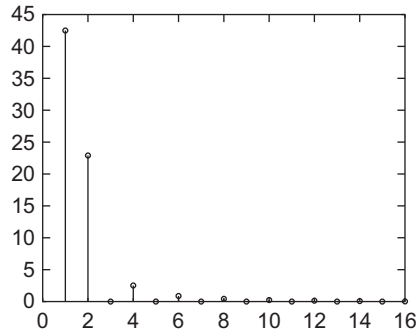


FIGURE 4.3-4

MATLAB plot of DCT $X_C(k)$.

where we also calculate the FFT magnitude for comparison purposes. The resulting plots are shown in Figures 4.3-2 through 4.3-4. We see that for this highly asymmetric signal the DCT is a much more efficient representation than is the DFT.

Some 1-D DCT Properties

1. Linearity:

$$ax + by \Leftrightarrow aX_C + bY_C.$$

2. Energy conservation:

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{2N} \sum_{k=0}^{N-1} w(k) |X_C(k)|^2. \quad (4.3-6)$$

3. Symmetry:**(a) General case:**

$$x^*(n) \Leftrightarrow X_C^*(k).$$

(b) Real-valued case:

$$\text{real } x(n) \Rightarrow \text{real } X_C(k).$$

4. Eigenvectors of unitary DCT:

Define the column vector $\mathbf{x} \triangleq [x(0), x(1), \dots, x(N-1)]^T$ and define the matrix \mathbf{C} with elements

$$C_{k',n'} \triangleq \begin{cases} \sqrt{\frac{1}{N}}, & k' = 1, \\ \sqrt{\frac{2}{N}} \cos \frac{\pi}{2N} (k' - 1)(2n' - 1), & 1 < k' \leq N. \end{cases}$$

Then the vector $\mathbf{y} = \mathbf{C}\mathbf{x}$ is the *unitary* DCT, whose elements are given as

$$\mathbf{y} \triangleq \left[\sqrt{\frac{1}{N}} X_C(0), \sqrt{\frac{2}{N}} X_C(1), \dots, \sqrt{\frac{2}{N}} X_C(N-1) \right]^T.$$

A *unitary matrix* is one whose inverse is the same as the transpose $\mathbf{C}^{-1} = \mathbf{C}^T$. For the unitary DCT, we have

$$\mathbf{x} = \mathbf{C}^T \mathbf{y},$$

and for the energy balance equation,

$$\begin{aligned} \mathbf{x}^T \mathbf{x} &= \mathbf{y} \mathbf{C} \mathbf{C}^T \mathbf{y} \\ &= \mathbf{y}^T \mathbf{y}, \end{aligned}$$

a slight modification on the DCT Parseval equation of (4.3–6). So the unitary DCT preserves the energy of the signal x .

It turns out that the eigenvectors of the unitary DCT are the same as those of the symmetric tridiagonal matrix [3],

$$\mathbf{Q} = \begin{bmatrix} 1-\alpha & -\alpha & 0 & \cdots & \cdots & 0 \\ -\alpha & 1 & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & -\alpha & \ddots & \vdots \\ \vdots & 0 & -\alpha & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 & -\alpha \\ 0 & \cdots & \cdots & 0 & -\alpha & 1-\alpha \end{bmatrix},$$

and this holds true for arbitrary values of the parameter α .

We can relate this matrix \mathbf{Q} to the inverse covariance matrix of a 1-D first-order stationary Markov random sequence [4], with correlation coefficient ρ , necessarily satisfying $|\rho| < 1$,

$$\mathbf{R}^{-1} = \frac{1}{\beta^2} \begin{bmatrix} 1 - \rho\alpha & -\alpha & 0 & \cdots & \cdots & 0 \\ -\alpha & 1 & -\alpha & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & -\alpha & \ddots & \vdots \\ \vdots & 0 & -\alpha & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 & -\alpha \\ 0 & \cdots & \cdots & 0 & -\alpha & 1 - \rho\alpha \end{bmatrix},$$

where $\alpha \triangleq \rho/(\rho + 1)$ and $\beta^2 \triangleq (1 - \rho^2)/(1 + \rho^2)$. The actual covariance matrix of the Markov random sequence is

$$\mathbf{R} = \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \rho^2 & \ddots & \vdots \\ \rho^2 & \rho & 1 & \rho & \ddots & \rho^3 \\ \rho^3 & \rho^2 & \rho & \ddots & \ddots & \rho^2 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \rho \\ \rho^{N-1} & \cdots & \rho^3 & \rho^2 & \rho & 1 \end{bmatrix},$$

with corresponding first-order difference equation

$$x(n) = \rho x(n-1) + w(n).$$

It can further be shown that when $\rho \simeq 1$, the matrix $\mathbf{Q} \simeq \beta^2 \mathbf{R}^{-1}$, so that their eigenvectors approximate each other too. Because the eigenvectors of a matrix and its inverse are the same, we know that the unitary DCT basis vectors approximate the Karhunen–Loeve expansion [4], with basis vectors given as the solution to the matrix-vector equation

$$\mathbf{R}\Phi = \Lambda\Phi,$$

and corresponding Karhunen–Loeve transform (KLT) given by

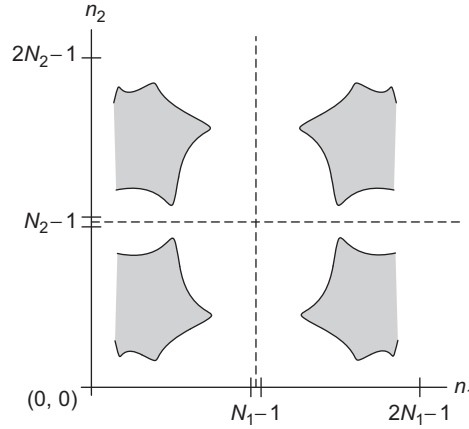
$$\mathbf{y} = \Phi^\dagger \mathbf{x}.$$

Thus, the 1-D DCT of a first-order Markov random vector of dimension N should be close to the KLT of x when its correlation coefficient $\rho \simeq 1$. This ends the review of the 1-D DCT.

Symmetric Extension in 2-D DCT

Since the 2-D DCT

$$X_C(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 4x(n_1, n_2) \cos \frac{\pi k_1}{2N_1} (2n_1 + 1) \cos \frac{\pi k_2}{2N_2} (2n_2 + 1)$$

**FIGURE 4.3-5**

An illustration of the 2-D symmetric extension used in the DCT.

is just the separable operator resulting from application of the 1-D DCT along first one dimension and then the other, the order being immaterial, we can easily extend the 1-D DCT properties to the 2-D case. In terms of the connection of the 2-D DCT with the 2-D DFT, we thus see that we must symmetrically extend in, say, the horizontal direction and then symmetrically extend that result in the vertical direction. The resulting symmetric function (extension) becomes

$$y(n_1, n_2) \triangleq x(n_1, n_2) + x(n_1, 2N_2 - 1 - n_2) + x(2N_1 - 1 - n_1, n_2) + x(2N_1 - 1 - n_1, 2N_2 - 1 - n_2),$$

which is sketched in Figure 4.3-5, where we note that the symmetry is about the lines $N_1 - \frac{1}{2}$ and $N_2 - \frac{1}{2}$. Then from (4.3-5), it follows that 2-D DCT is given in terms of the $N_1 \times N_2$ -point DFT as

$$X_C(k_1, k_2) = \begin{cases} W_{2N_1}^{k_1/2} W_{2N_2}^{k_2/2} Y(k_1, k_2), & (k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1], \\ 0, & \text{else.} \end{cases}$$

Comments

1. We see that both the 1-D and 2-D DCTs only involve real arithmetic for real-valued data, and this may be important in some applications.
2. The symmetric extension property can be expected to result in fewer high-frequency coefficients in DCT with respect to DFT. Such would be expected for lowpass data, since there would often be a jump at the four edges of the $N_1 \times N_2$ period of the corresponding periodic sequence $\tilde{x}(n_1, n_2)$, which is not consistent with small high-frequency coefficients in the DFS or DFT. Thus the DCT is attractive for lossy data storage applications, where the exact value of the data is not of paramount importance.

3. The DCT can be used for a symmetrical type of filtering with a symmetrical filter. (See end-of-chapter problem 16.)
4. 2-D DCT properties are easy generalizations of 1-D DCT properties reviewed in this section.

4.4 SUBBAND/WAVELET TRANSFORM⁵

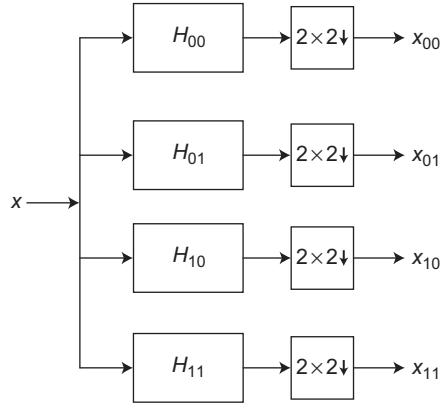
The DCT transform is widely used for compression, in which case it is almost always applied as a *block transform*. In a block transform, the data are scanned, a number of lines at a time, and the data are then mapped into a sequence of blocks. These blocks are then operated upon by 2-D DCT. In the image compression standard JPEG, the DCT is used with a block size of 8×8 . Subband/wavelet transforms (SWTs) can be seen as a generalization of such block transforms, a generalization that allows the blocks to overlap. SWTs make direct use of decimation and expansion and LSI filters. In Example 2.3–1 of Chapter 2, we saw that ideal rectangular filters could be used to decompose the $2\pi \times 2\pi$ unit cell of the frequency domain into smaller regions. Decimation could then be used on these smaller regions to form component signals at lower spatial sample rates. The collection of these lower sample rate signals then would be equivalent to the original high-rate signal. However, being separate signals, now they are often amenable to processing targeted to their reduced frequency range, and this has been found useful in many applications.

Ideal Filter Case

To be specific, consider the four-subband decomposition shown in Figure 4.4–1. Using ideal rectangular filters, we can choose H_{00} to pass the LL subband $\{|\omega_1| \leq \pi/2, |\omega_2| \leq \pi/2\}$ and reject other frequencies in $[-\pi, +\pi]^2$. Then set H_{10} to pass the HL subband $\{\pi/2 < |\omega_1|, |\omega_2| \leq \pi/2\}$, set H_{01} to pass the LH subband $\{|\omega_1| \leq \pi/2, \pi/2 < |\omega_2|\}$, and finally H_{11} to pass the HH subband $\{\pi/2 < |\omega_1|, \pi/2 < |\omega_2|\}$, all defined as subsets of $[-\pi, +\pi]^2$. Then after 2×2 decimation, the four-subband signals x_{00}, x_{10}, x_{01} , and x_{11} will contain all the information from the input signal x , each with a four-times lower spatial sample rate. For example, using (2.3–1) from Section 2.3, we have for the 2×2 decimation case, where $M_1 = M_2 = 2$,

$$X_{ij}(\omega_1, \omega_2) = \frac{1}{4} \sum_{k_1=0}^1 \sum_{k_2=0}^1 H_{ij} \left(\frac{\omega_1 - 2\pi k_1}{2}, \frac{\omega_2 - 2\pi k_2}{2} \right) X \left(\frac{\omega_1 - 2\pi k_1}{2}, \frac{\omega_2 - 2\pi k_2}{2} \right). \quad (4.4-1)$$

⁵What we here call the subband/wavelet transform (SWT) is usually referred to as either the discrete wavelet transform (DWT) or the subband transform.

**FIGURE 4.4–1**

An illustration of a 2×2 rectangular SWT.

For the case $i = j = 0$, and noting the filter H_{00} has unity gain in its $[-\pi/2, +\pi/2]^2$ passband, we have the LL subband

$$X_{00}(\omega_1, \omega_2) = \frac{1}{4}X\left(\frac{\omega_1}{2}, \frac{\omega_2}{2}\right) \text{ on } [-\pi, +\pi]^2, \quad (4.4-2)$$

because the other three terms in (4.4–1), with i and j not both 0, are all zero in the frequency domain unit cell $[-\pi, +\pi]^2$. Thus we can write the LL subband signal x_{00} as⁶

$$x_{00}(n_1, n_2) = \text{IFT} \left\{ \frac{1}{4}X(\omega_1/2, \omega_2/2) \text{ on } [-\pi, +\pi]^2 \right\}.^7$$

Similarly, regarding the LH subband x_{01} , we can say that

$$x_{01}(n_1, n_2) = \text{IFT} \left\{ \frac{1}{4}X(\omega_1/2, \omega_2/2 - \pi) \text{ on } [-\pi, +\pi]^2 \right\},$$

and so forth for x_{10} and x_{11} . Thus we have the following frequency domain expressions, valid in the unit frequency cell $[-\pi, +\pi]^2$ and then periodically repeated

⁶Note that $\frac{1}{4}X(\omega_1/2, \omega_2/2)$ is not periodic with period $2\pi \times 2\pi$ and so is not a complete expression for the FT. Still, over the fundamental period $[-\pi, +\pi]^2$, it is correct. The complete expression for the FT just repeats this expression, periodically, outside $[-\pi, +\pi]^2$.

⁷We write the function to be inverse Fourier transformed in the unconventional way $\frac{1}{4}X(\omega_1/2, \omega_2/2)$ on $[-\pi, +\pi]^2$ to emphasize that $\frac{1}{4}X(\omega_1/2, \omega_2/2)$ by itself is not a Fourier transform, as mentioned in the preceding note.

elsewhere:

$$\begin{aligned} X_{00}(\omega_1, \omega_2) &= \frac{1}{4}X(\omega_1/2, \omega_2/2), \\ X_{01}(\omega_1, \omega_2) &= \frac{1}{4}X(\omega_1/2, \omega_2/2 - \pi), \\ X_{10}(\omega_1, \omega_2) &= \frac{1}{4}X(\omega_1/2 - \pi, \omega_2/2), \text{ and} \\ X_{11}(\omega_1, \omega_2) &= \frac{1}{4}X(\omega_1/2 - \pi, \omega_2/2 - \pi). \end{aligned}$$

A sketch of an isotropic X and its corresponding LH subband X_{01} are shown in Figure 4.4–2 and Figure 4.4–3, respectively. Note that the LH subband has captured the part of X that is low frequency in ω_1 and high frequency in ω_2 .

To complete this 2×2 rectangular SWT, we must show the inverse transform to reconstruct the original signal x from these four-subband signals x_{00}, x_{01}, x_{10} , and x_{11} . The inverse transform system diagram is shown in Figure 4.4–4. With reference to (2.3–4) from Chapter 2, we can write the Fourier transform output of each filter as

$$G_{kl}(\omega_1, \omega_2)X_{kl}(2\omega_1, 2\omega_2), \text{ for } k, l = 0, 1.$$

Combining these results, we obtain the overall equation for transform followed by inverse transform as

$$X(\omega_1, \omega_2) = \sum_{k,l=0,1} G_{kl}(\omega_1, \omega_2)X_{kl}(2\omega_1, 2\omega_2). \quad (4.4-3)$$

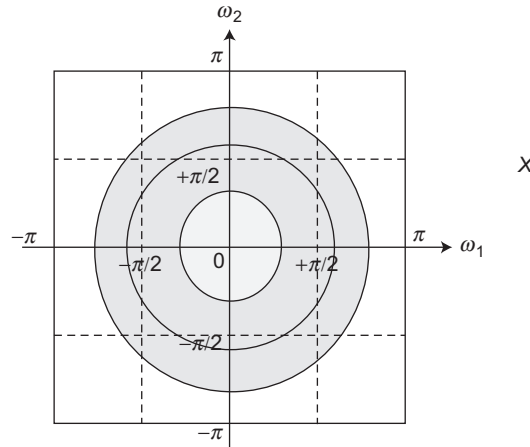
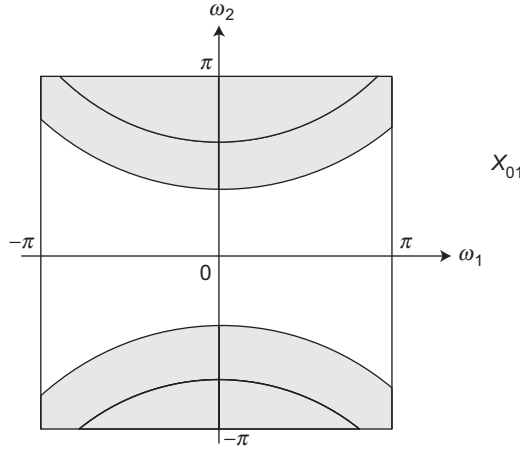
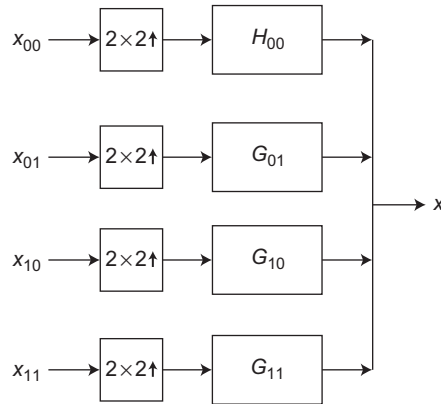


FIGURE 4.4–2

Fourier transform X of isotropic signal.

**FIGURE 4.4-3**

Fourier transform X_{01} of LH subband of isotropic signal.

**FIGURE 4.4-4**

An illustration of a 2×2 rectangular subband/wavelet inverse transform.

Now if we set the gains of the ideal synthesis filters $G_{kl} = 4H_{kl}$, and define their frequency-domain regions of support identical to those of the analysis filters H_{kl} , we can obtain perfect reconstruction as follows. First, note that each of the four $X_{kl}(2\omega_1, 2\omega_2)$ terms are periodic with the smaller period $\pi/2 \times \pi/2$, which matches exactly the support regions of the ideal bandpass filters G_{kl} . We can thus consider the separate regions, one at a time. First, consider the LL subband region $[-\pi/2, +\pi/2]^2$ where G_{00} has its support and $X_{00}(2\omega_1, 2\omega_2) = \frac{1}{4}X(\omega_1, \omega_2)$ from (4.4-3). Now the other three terms in the preceding reconstruction equation are zero here; thus, for this LL subband region, we have the perfect reconstructed output $X(\omega_1, \omega_2)$, as desired.

Next, consider the LH subband region $[-\pi/2, +\pi/2] \times [\pi/2, \pi]$, where G_{01} has its support, and we use (4.4-3) to find $X_{01}(2\omega_1, 2\omega_2) = \frac{1}{4}X(\omega_1, \omega_2 - \pi)$ on $[-\pi/2, +\pi/2]^2$ and periodically repeated outside. So we need its first vertical alias valid in the region $[-\pi/2, +\pi/2] \times [\pi/2, \pi]$, which we get by adding 2π to the argument ω_2 in (4.4-3) to first get $X_{01}(\omega_1, \omega_2) = \frac{1}{4}X(\omega_1/2, (\omega_2 + 2\pi)/2 - \pi) = \frac{1}{4}X(\omega_1/2, \omega_2/2)$, valid in the region $[-\pi, +\pi] \times [\pi, 3\pi]$; therefore, $X_{01}(2\omega_1, 2\omega_2) = \frac{1}{4}X(\omega_1, \omega_2)$ in the region $[-\pi/2, +\pi/2] \times [\pi/2, 3\pi/2]$, which is the same as the union of the two regions $[-\pi/2, +\pi/2] \times [\pi/2, \pi]$ and $[-\pi/2, +\pi/2] \times [-\pi, -\pi/2]$ by the necessary $2\pi \times 2\pi$ periodicity. Reference to Figure 4.4-3 may help here. Thus for the LH subband region where G_{01} is nonzero, we have $X_{01}(2\omega_1, 2\omega_2) = \frac{1}{4}X(\omega_1, \omega_2 - \pi + \pi) = \frac{1}{4}X(\omega_1, \omega_2)$ and thereby get perfect reconstruction with a passband gain of 4. The perfect reconstruction for the other two subband regions HL $[\pi/2, \pi] \times [-\pi/2, +\pi/2]$ and HH $[\pi/2, \pi] \times [-\pi/2, +\pi/2]$ follows similarly.

Putting the four results together, we have

$$X(\omega_1, \omega_2) = \begin{cases} X(\omega_1, \omega_2), & |\omega_1| \leq \pi/2, |\omega_2| \leq \pi/2, \\ X(\omega_1, \omega_2), & |\omega_1 - \pi| < \pi/2, |\omega_2| \leq \pi/2, \\ X(\omega_1, \omega_2), & |\omega_1| \leq \pi/2, |\omega_2 - \pi| < \pi/2, \\ X(\omega_1, \omega_2), & |\omega_1 - \pi| < \pi/2, |\omega_2 - \pi| < \pi/2, \end{cases}$$

$$= X(\omega_1, \omega_2) \quad \checkmark.$$

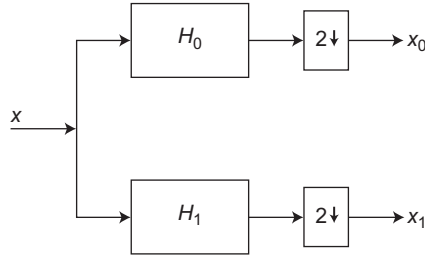
We can regard this as an ideal SWT in the sense that the Fourier transform is an ideal frequency transform, while the DFT and DCT are more practical ones; that is, they involve a finite amount of computation and do not have ideal filter shapes. We can substitute nonideal filters for the preceding ideal filter case and get a more practical type of SWT. The resulting transform, while not avoiding aliasing error in the subband signals themselves, does manage to cancel out this aliasing error in the *reconstructed* or inverse transformed signal x . To distinguish the two, we can call the preceding an *ideal* SWT and a practical type, with finite order filters, just an SWT. Of course, there is no unique SWT, as it will depend on the filters used.

1-D SWT with Finite Order Filters

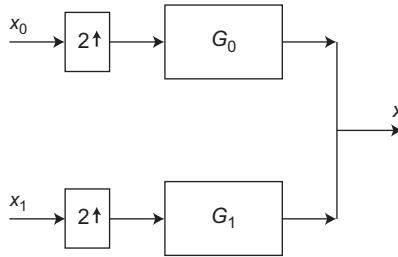
We can avoid the need for ideal filters if we can accept some aliasing error in the subbands. We first review the 1-D case for a two-channel filter bank. Then we will construct the required 2-D filters as separable products. A simple two-channel 1-D SWT is shown in Figure 4.4-5. If we construct the 1-D filters to cancel out any aliasing in the reconstruction, then the same will be true of the separable 2-D transform. Using the 1-D theory, we find

$$X_0(\omega) = \frac{1}{2} [H_0(\omega/2)X(\omega/2) + H_0(\omega/2 - \pi)X(\omega/2 - \pi)] \quad \text{and}$$

$$X_1(\omega) = \frac{1}{2} [H_1(\omega/2)X(\omega/2) + H_1(\omega/2 - \pi)X(\omega/2 - \pi)].$$

**FIGURE 4.4-5**

A system diagram for a two-channel 1-D SWT.

**FIGURE 4.4-6**

An illustration of the 1-D ISWT.

The inverse 1-D SWT is shown in Figure 4.4-6. From this figure, the reconstructed signal (inverse transform) is given as

$$\begin{aligned}\hat{X}(\omega) &= G_0(\omega)X_0(2\omega) + G_1(\omega)X_1(2\omega) \\ &= \frac{1}{2} [G_0(\omega)H_0(\omega) + G_1(\omega)H_1(\omega)] X(\omega) \\ &\quad + \frac{1}{2} [G_0(\omega)H_0(\omega - \pi) + G_1(\omega)H_1(\omega - \pi)] X(\omega - \pi).\end{aligned}$$

Therefore, to cancel out aliasing, we need

$$G_0(\omega)H_0(\omega - \pi) + G_1(\omega)H_1(\omega - \pi) = 0, \quad (4.4-4)$$

and then to achieve perfect reconstruction, we additionally need

$$G_0(\omega)H_0(\omega) + G_1(\omega)H_1(\omega) = 2. \quad (4.4-5)$$

The first solution to this problem was the *quadrature mirror condition* [5, 6],

$$G_0(\omega) \triangleq H_1(\omega - \pi) \quad \text{and} \quad G_1(\omega) \triangleq -H_0(\omega - \pi), \quad (4.4-6)$$

which cancels out the aliasing term (4.4-4). In order to get a lowpass–highpass filter pair, we then set

$$H_1(\omega) = H_0(\omega - \pi), \quad (4.4-7)$$

where we see the “mirror symmetry,” as sketched in Figure 4.4-7.

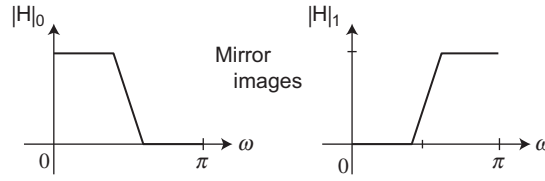
**FIGURE 4.4-7**

Illustration of magnitude responses of a quadrature magnitude filter (QMF) pair.

We can see that if H_0 is chosen as a lowpass filter, then H_1 will be highpass, and similarly the reconstruction filter G_0 will be lowpass and G_1 will be highpass. To get perfect reconstruction with zero delay, we then need

$$H_0^2(\omega) - H_0^2(\omega - \pi) = 2. \quad (4.4-8)$$

Often, a positive integer delay is inserted on the right-hand side of the condition (4.4-8) to allow causal filters to be used in both analysis (SWT) and synthesis (ISWT).

Example 4.4-1: Haar Filter Pair

Nearly the simplest subband/wavelet filter pair is obtained by forming the sum and difference of successive pairs of inputs, as can be accomplished with the so-called Haar filters

$$h_0(n) \triangleq \frac{1}{\sqrt{2}} [\delta(n) + \delta(n-1)] \quad \text{and} \quad h_1(n) \triangleq \frac{1}{\sqrt{2}} [\delta(n) - \delta(n-1)],$$

which agree with the mirror symmetry condition in (4.4-7). Then, with reference to Figure 4.4-5, we have the SWT transform functions

$$x_0(n) = \frac{1}{\sqrt{2}} [x(2n) + x(2n-1)] \quad \text{and} \quad x_1(n) = \frac{1}{\sqrt{2}} [x(2n) - x(2n-1)].$$

Next, following the synthesis condition in (4.4-6), we obtain

$$g_0(n) = \frac{1}{\sqrt{2}} [\delta(n) + \delta(n-1)] \quad \text{and} \quad g_1(n) = \frac{1}{\sqrt{2}} [-\delta(n) + \delta(n-1)].$$

Taking the Fourier transforms of these filters and inserting into the left-hand side of (4.4-8), we get

$$\begin{aligned} H_0^2(\omega) - H_0^2(\omega - \pi) &= \frac{1}{2} (1 + 2e^{-j\omega} + e^{-j2\omega}) - \frac{1}{2} (1 - 2e^{-j\omega} + e^{-j2\omega}) \\ &= \frac{1}{2} 4e^{-j\omega} \\ &= 2e^{-j\omega}, \end{aligned}$$

indicating perfect reconstruction is possible, but with a delay of one sample.

An alternative and preferable formulation of the alias cancellation for 1-D SWT was found by Simoncelli and Adelson [7]. In their approach,

$$G_0(\omega) \triangleq H_0(-\omega) \quad \text{and} \quad G_1(\omega) \triangleq H_1(-\omega), \quad (4.4-9)$$

which results in the reconstruction formula

$$\begin{aligned} \hat{X}(\omega) &= \frac{1}{2} [H_0(\omega)H_0(-\omega) + H_1(\omega)H_1(-\omega)]X(\omega) + \frac{1}{2} [H_0(\omega - \pi)H_0(-\omega) \\ &\quad + H_1(\omega - \pi)H_1(-\omega)]X(\omega - \pi) \\ &= \frac{1}{2} [|H_0(\omega)|^2 + |H_1(\omega)|^2]X(\omega) + \text{alias terms}, \end{aligned}$$

since the h_i are assumed real. Upon setting

$$H_0(\omega) = H(-\omega) \quad \text{and} \quad H_1(\omega) = e^{-j\omega}H(\omega + \pi), \quad (4.4-10)$$

again the alias terms cancel out. Here, $H(\omega)$ is chosen as a lowpass filter designed to achieve

$$\begin{aligned} |H_0(\omega)|^2 + |H_1(\omega)|^2 &= \\ |H(\omega)|^2 + |H(\omega + \pi)|^2 &= 2. \end{aligned} \quad (4.4-11)$$

Example 4.4-2: Alternative Haar Filter Pair

If we follow this alternative method for the Haar lowpass filter $h_0(n) = \frac{1}{\sqrt{2}} [\delta(n) + \delta(n-1)]$, we get the alternative highpass filter $h_1(n) = \frac{1}{\sqrt{2}} [-\delta(n) + \delta(n-1)]$ via (4.4-10) and then via (4.4-9), the pair of noncausal reconstruction filters

$$g_0(n) = \frac{1}{\sqrt{2}} [\delta(n) + \delta(n+1)] \quad \text{and} \quad g_1(n) = \frac{1}{\sqrt{2}} [-\delta(n) + \delta(n+1)].$$

Upon insertion into (4.4-11), we get perfect reconstruction with zero delay for this noncausal pair. ■

Note the similarity in the two Haar SWTs in the last two examples. In fact, if we put a delay of one sample into the noncausal reconstruction (ISWT) in this example, they become equivalent solutions. Unfortunately for filter orders higher than one, exact reconstructions are not possible, and approximation to the reconstruction formulas is thus necessary. Some methods of subband/wavelet filter design will be discussed in Chapter 5.

2-D SWT with Finite Impulse Response (FIR) Filters

We can apply the just-reviewed 1-D SWT theory separably to the horizontal and vertical axes, resulting in four subbands $X_{ij}(\omega_1, \omega_2)$, just as in (4.4-1). Since the 2-D separable case can be thought of as the sequential application of the earlier 1-D SWT

to the rows and columns, we know immediately that the corresponding separable 2-D SWT will achieve alias cancellation and perfect reconstruction also,

$$H_{ij}(\omega_1, \omega_2) = H_i(\omega_1)H_j(\omega_2) \quad \text{and} \quad G_{ij}(\omega_1, \omega_2) = G_i(\omega_1)G_j(\omega_2).$$

Due to finite order filters, though, equations such as (4.4-2) will not hold and alias error will be present in the subbands X_{ij} . However, if we use as a basis for our separable filtering, filters such as the preceding ones that have the property of canceling out aliasing in the reconstruction, then *inverse* SWT (ISWT) exists as indicated in (4.4-3). Depending on the filters used and their length, the amount of aliasing in the subbands may be significant. Filter designs for SWT can ameliorate this problem, mainly by going to longer filters.

Relation of SWT to DCT

Consider an 8×8 rectangular SWT. It would have 64 subbands, with center frequencies evenly distributed over the frequency unit cell. Now consider applying the DCT to the same data, but blockwise using 8×8 DCTs. We can easily identify each set of DCT coefficients with one of the subbands. In fact, if we were to use the DCT basis functions, reflected through the origin as the subband/wavelet filters,

$$h_{kl}(n_1, n_2) = 4 \cos \frac{\pi k}{2N_1} (-2n_1 + 1) \cos \frac{\pi l}{2N_2} (-2n_2 + 1),$$

then the subband values would be exactly the sequence of DCT coefficients at that frequency k, l . We can thus see the SWT as a generalization of the block DCT transform, wherein the basis functions (subband filter impulse responses) are allowed to overlap in space.

Relation of SWT to Wavelets

Wavelet theory is essentially the continuous-time theory that corresponds to dyadic subband transforms—i.e., those where the L (LL) subband is recursively split over and over. Wavelet analysis of a continuous-time signal begins as follows. Let $f(t) \in L^2$ (L^2 being the space of square integrable functions $\int_{-\infty}^{+\infty} |f(t)|^2 dt < \infty$), specify a *mother wavelet* $\psi(t)$ as some

$$\psi(t) \in L^1 \text{ satisfying } \int_{-\infty}^{+\infty} \psi(t) dt = 0,$$

i.e., a specified highpass function. We then define a *continuous wavelet transform* (CWT) of f as [8, 9]

$$\gamma(s, \tau) \triangleq \int_{-\infty}^{+\infty} f(t) \psi_{s,\tau}(t) dt,$$

with wavelets $\psi_{s,\tau}(t) \triangleq \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right)$. The positive parameter s is called the *scale*, and τ is called the *delay*. We can see that if ψ is concentrated in time, a CWT gives a time-varying analysis of f into scales, or resolutions, where increasing the scale parameter s provides a lower resolution analysis. You might think that f is being overanalyzed here, and indeed it is. Wavelet theory shows that a discrete version of this analysis suffices to describe the function. A *discrete wavelet transform* (still for continuous-time functions) is defined via the discrete family of wavelets [6, 11]:

$$\begin{aligned}\psi_{k,l}(t) &\triangleq \frac{1}{\sqrt{s_0^k}} \psi\left(\frac{t-l\tau_0 s_0^k}{s_0^k}\right), \\ &= 2^{-k/2} \psi(2^{-k}t - l), \text{ with } s_0 = 2 \text{ and } \tau_0 = 1,\end{aligned}$$

as

$$\gamma(k,l) \triangleq \int_{-\infty}^{+\infty} f(t) \psi_{k,l}(t) dt.$$

This analysis can be inverted as follows:

$$f(t) = \sum_{k,l} \gamma(k,l) \psi_{k,l}(t), \quad (4.4-12)$$

called an *inverse discrete wavelet transform* (IDWT), thus creating the analysis–synthesis pair that justifies the use of the word “transform.” Now the k index denotes scale, so that as k increases, the scale gets larger and larger, while the l index denotes delay or position of the scale information on the time axis. The values of the DWT $\gamma(k,l)$ are also called *wavelet coefficients* of the continuous-time function f . Here, both indices k,l are doubly infinite (i.e., $-\infty < k,l < +\infty$) in order to span the full range of scales and delays.

Rewriting (4.4-12) as a sum over scales,

$$f(t) = \sum_k \left[\sum_l \gamma(k,l) \psi_{k,l}(t) \right], \quad (4.4-13)$$

we can break the scales up into two parts,

$$\begin{aligned}f(t) &= \sum_{k < k_0} \left[\sum_l \gamma(k,l) \psi_{k,l}(t) \right] + \sum_{k \geq k_0} \left[\sum_l \gamma(k,l) \psi_{k,l}(t) \right], \\ &= \text{low-scale (frequency) part} + \text{fine-scale (frequency) part},\end{aligned}$$

which is reminiscent of a SWT that we have defined only for discrete-time functions earlier in this chapter. In fact, wavelet theory shows that the two theories fit together in the following sense. If a continuous-time function is wavelet analyzed with a DWT at a fine scale, then the wavelet coefficients at the different scales are related by a SWT. Going a bit further, to each DWT there corresponds a dyadic SWT that recursively

calculates the wavelet coefficients. Unfortunately, there are SWTs that do not correspond to wavelet transforms due to lack of convergence as the scale gets larger and larger. This is the so-called *regularity* problem. However, it can be generally said that most SWTs currently being used have good enough regularity properties, given the rather small number of scales that are used both for image analysis and for image coding, typically less than seven. More information on wavelets and wavelet transforms can be obtained from many sources, including the classic works of [8, 9].

Before leaving the topic of wavelets, notice that the scale parameter k in (4.4–12) never gets to $-\infty$, as the infinite sum is really just the limit of finite sums. However, this is OK because there cannot be any “dc” value or constant in any function $f(t) \in L^2$ over the infinite interval $(-\infty, +\infty)$, since we require that the mother wavelet and, as a result, the actual wavelet basis functions, have zero mean, i.e., $\int_{-\infty}^{+\infty} \psi(t) dt = 0$.

4.5 FAST TRANSFORM ALGORITHMS

The DFT and DCT are widely used in image and video signal processing, due in part to the presence of fast algorithms for their calculation. These fast algorithms are largely inherited from the 1-D case due to the separable nature of these transforms.

Fast DFT Algorithm

Here, we briefly look at efficient algorithms to calculate the DFT. We cover the so-called row–column approach, which expresses the 2-D DFT as a series of 1-D transforms. To understand this method we start out with

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2},$$

valid for $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$, and bring the sum over n_1 inside, to obtain

$$\begin{aligned} &= \sum_{n_2=0}^{N_2-1} \left[\sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} \right] W_{N_2}^{n_2 k_2} \\ &= \sum_{n_2=0}^{N_2-1} X(k_1; n_2) W_{N_2}^{n_2 k_2} \end{aligned} \quad (4.5-1)$$

upon defining the row-transforms $X(k_1; n_2)$ as

$$X(k_1; n_2) \triangleq \sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1},$$

for each row n_2 of the original data array x . If we store these values back in the same row, then we see that (4.5–1) is then just the 1-D column DFTs of this intermediate data array. If the data are stored sequentially on a tape or disk, it may be advantageous to perform a data transposition step between the column and row transforms to minimize access to this secondary slower storage [1]. When there is sufficient random access storage, such an intermediate step is not necessary. If we use an in-place 1-D FFT routine to implement the row and column transforms, then the amount of RAM needed would be determined by the size of the data array x with support $N_1 \times N_2$.

Computation

Here, we assume that both N_1 and N_2 are powers of 2.

Step 1: We use the Cooley-Tukey 1-D FFT algorithm to do the row transforms, either *decimation in time* (DIT) or *decimation in frequency* (DIF) with $N_2 \cdot \frac{N_1}{2} \log_2 N_1$ complex multiplies and $N_2 \cdot N_1 \log_2 N_1$ complex additions.

Step 2: We do any necessary data mappings due to insufficient RAM (e.g., matrix transposition).

Step 3: We use the Cooley-Tukey 1-D FFT algorithm again to do the column transforms in $N_1 \cdot \frac{N_2}{2} \log_2 N_2$ complex multiplies and $N_1 \cdot N_2 \log_2 N_2$ complex additions.

The total amount of computation then comes to $\frac{N_1 N_2}{2} \log_2 N_1 N_2$ complex multiplications and $N_1 N_2 \log_2 N_1 N_2$ complex additions. These are the so-called *radix-2* algorithms. There are also *radix-4* algorithms for 1-D FFT and vector radix algorithms for 2-D FFT [1]. They both can offer some modest improvement over the radix-2 case considered here. For example, if N is a power of 4, then the radix-4 approach can save about 25% in computation.

Fast DCT Methods

We can employ a fast method of DFT calculation to the DCT. We would simply calculate the $2N$ -point DFT of the symmetrically extended sequence y . This would be an efficient $N \log_2 N$ method but would involve complex arithmetic. An alternative method involving only real arithmetic for a real-valued $x(n)$ has been obtained by [10]. They define the real and imaginary parts of the N -point DFT as

$$\cos\text{-DFT}_N(k) \triangleq \sum x(n) \cos \frac{2\pi nk}{N}$$

and

$$\sin\text{-DFT}_N(k) \triangleq \sum x(n) \sin \frac{2\pi nk}{N}.$$

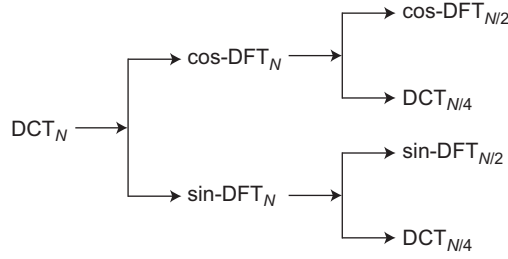


FIGURE 4.5–1

Tree diagram of fast DCT by [10].

Then they find

$$\begin{aligned}
 \text{DFT}_N(k) &= \cos\text{-DFT}_N(k) - j \sin\text{-DFT}_N(k), \\
 \text{DCT}_N(N) &= 0, \\
 \text{DCT}_N(-k) &= \text{DCT}_N(+k), \\
 \text{DCT}_N(2N - k) &= -\text{DCT}_N(k), \\
 \cos\text{-DFT}_N(N - k) &= \cos\text{-DFT}_N(k), \text{ and} \\
 \sin\text{-DFT}_N(N - k) &= -\sin\text{-DFT}_N(k),
 \end{aligned}$$

where in these equations, and others of this section, the DCT and DFT transform is not automatically set to zero outside its fundamental period. Using these relations and some others, the dependency tree of Figure 4.5–1 is developed [10].

The resulting computation for N , a positive integral power of two becomes approximately $\frac{N}{2} \log_2 N$ real multiplications and $\frac{N}{2} (3 \log_2 N - 2) + 1$ real additions (see [10] for details). The 2-D $N_1 \times N_2$ DCT, via a row–column method, would then require $N_1 \cdot \frac{N_2}{2} \log_2 N_2 + N_2 \cdot \frac{N_1}{2} \log_2 N_1$ real multiplications.

4.6 SECTIONED CONVOLUTION METHODS

Consider an image of size $M_1 \times M_2$, to be convolved with an FIR filter of size $L_1 \times L_2$, where usually we have $L_i < M_i, i = 1, 2$. In such a case, it is not efficient to directly implement the 2-D convolution in the DFT domain. However, we can first section the input into adjoining rectangular sections $x_{i,j}(n_1, n_2)$ of intermediate support, so that

$$x(n_1, n_2) = \sum_{i,j} x_{i,j}(n_1, n_2).$$

Then we can write the convolution $y = h * x$ as

$$y(n_1, n_2) = \sum_{i,j} h(n_1, n_2) * x_{i,j}(n_1, n_2),$$

where each section output $y_{i,j}(n_1, n_2) \triangleq h(n_1, n_2) * x_{i,j}(n_1, n_2)$ is of much smaller extent, and so can be efficiently calculated via DFT methods. In the 2-D *overlap-and-add* (O&A) method [2], we select the section size as $(N_1 - L_1 + 1) \times (N_2 - L_2 + 1)$ and the DFT size as $N_1 \times N_2$ so as to avoid spatial aliasing. The section outputs $y_{i,j}$ then overlap in vertical strips of width L_1 and horizontal strips of height L_2 .

The total number of sections for the image N_s is given as

$$N_s = \left\lceil \frac{M_1}{N_1 - L_1 + 1} \right\rceil \left\lceil \frac{M_2}{N_2 - L_2 + 1} \right\rceil,$$

where $\lceil \cdot \rceil$ indicates the *greatest integer function*. The computation needed per section is two DFTs and $N_1 N_2$ complex multiplies and additions. Using row-column FFTs and assuming that N_1 and N_2 are integral powers of 2, we obtain the total requirement as

$$\left\lceil \frac{M_1}{N_1 - L_1 + 1} \right\rceil \left\lceil \frac{M_2}{N_2 - L_2 + 1} \right\rceil [2N_1 N_2 \log_2(N_1 N_2) + 2N_1 N_2] \text{ real multiplications.}$$

Now the direct computation, assuming x and h are real valued, is $M_1 M_2 L_1 L_2$. We can offer the following comparisons. For $L_1 = L_2 \geq 5$, the O&A method has less computation according to these formulas, and for $L_1 = L_2 \geq 10$, the O&A method may be actually preferable, considering the overhead of implementing the required DFTs. In the literature on block filtering, typically $N_1 = N_2 = 64$ or 128 .

CONCLUSIONS

The DFS, DFT, and DCT can easily be extended to two dimensions as separable operators. The familiar concepts of circular convolution and now spatial aliasing were revisited again, due to the sampling in frequency space. We investigated the SWT and interpreted it as an extension to overlapping bases functions of the DCT. Fast algorithms for both the DFT and DCT were presented relying on the so-called row-column method that exploits operator separability. Sectioned convolution allows “linear convolution” results to be achieved via DFT-based circular convolution. More on the 2-D DFT, including a special DFT for general sampling lattices, can be found in Chapter 2 of Dudgeon and Mersereau [11].

PROBLEMS

1. Consider the rectangularly periodic sequence $\tilde{x}(n_1, n_2)$, with period $N_1 \times N_2$, and express its Fourier transform in terms of impulse functions. Relate your result to the corresponding DFS $\tilde{X}(k_1, k_2)$.

2. Show that the sum and product of two periodic functions \tilde{x} and \tilde{y} that have the same periods are periodic with the common period.
3. Find the N -point DFT of $x(n) = u(n) - u(n - N)$ and then the same size DFT of $x(n) = u(n) - u(n - N + 1)$.
4. Let the signal $x(n_1, n_2)$ have support $[0, N_1 - 1] \times [0, N_2 - 1]$. Express the $N_1 \times N_2$ point DFS or DFT, *as appropriate*, of each of the following in terms of $X(\omega_1, \omega_2)$, the Fourier transform of x :
 - (a) $x[(n_1)_{N_1}, (n_2)_{N_2}]$
 - (b) $\sum_{\text{all } k_1} \sum_{\text{all } k_2} x(n_1 - k_1 N_1, n_2 - k_2 N_2)$ (Remember, k_1 and k_2 are integers.)
 - (c) $x[(N_1 - n_1)_{N_1}, (N_2 - n_2)_{N_2}]$
 - (d) $x(N_1 - 1 - n_1, N_2 - 1 - n_2)$
5. Find the DFT of $x(n_1, n_2) = \delta(n_1, n_2) + 2\delta(n_1 - 1, n_2) + \delta(n_1 - 2, n_2)$.
6. Let an image $x(n_1, n_2)$ have support $n_1, n_2 := 0, \dots, N - 1$. Assuming the image is real valued, its DFT $X(k_1, k_2)$ will display certain symmetry properties. In particular, show that $X(k_1, k_2) = X^*(N - k_1, N - k_2)$ for $k_1, k_2 := 0, \dots, N - 1$.
7. Show that the 2-D DFT can be written in matrix-product form as the triple product

$$\mathbf{X} = \mathbf{F} \mathbf{x} \mathbf{F}^T,$$

for image data $\{x(n_1, n_2), 0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1\}$ entered into the $N_1 \times N_2$ matrix \mathbf{x} as

$$\mathbf{x}_{i,j} = x(i - 1, j - 1) \text{ for } 1 \leq i \leq N_1, 1 \leq j \leq N_2,$$

by choosing appropriate entries for the matrix \mathbf{F} . Show how the DFT is stored in the matrix \mathbf{X} in detail. This is a consequence of the separable operator nature of the DFT. Interpret this result in terms of row-DFTs and column-DFTs.

8. Prove Parseval's theorem for the $N_1 \times N_2$ -point DFT,

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) y^*(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) Y^*(k_1, k_2).$$

Your proof is to be direct from the definitions of DFT and IDFT and should not use any other properties.

9. Find the 2-D DFT of signal x , with support $[0, N_1 - 1] \times [0, N_2 - 1]$, given on its support as

$$x(n_1, n_2) = 10 + 2 \cos(2\pi 5 n_1 / N_1) + 5 \sin(2\pi 8 n_2 / N_2).$$

Assume that $N_1 > 5$ and $N_2 > 8$.

10. Prove the correctness of DFT property 2 (Section 4.2) directly in the 2-D DFT transform domain, using the DFT circular shift or delay property 5.

11. Prove the following 2-D DFT properties. Your proof should be direct (i.e., should not make use of other DFT or DFS properties). Take the DFT size to be $N_1 \times N_2$.
- (a) $x^*(n_1, n_2) \Leftrightarrow X^*[(-k_1)_{N_1}, (-k_2)_{N_2}]I_{N_1 \times N_2}(k_1, k_2)$.
- (b) $x^*[(-n_1)_{N_1}, (-n_2)_{N_2}]I_{N_1 \times N_2}(n_1, n_2) \Leftrightarrow X^*(k_1, k_2)$.
12. Start with a general signal $s(n_1, n_2)$ and take its Fourier transform to get $S(\omega_1, \omega_2)$. Then sample this Fourier transform at $\omega_1 = \frac{2\pi k_1}{256}$ and $\omega_2 = \frac{2\pi k_2}{256}$, where $k_1, k_2 : 0, \dots, 255$ to get $S(k_1, k_2)$. Find an expression for $\text{IDFT}\{S(k_1, k_2)\}$ in terms of the original $s(n_1, n_2)$. This is called *spatial aliasing*.
13. Fill in the details in going from (4.2–7) to (4.2–8), thus deriving the consequences of sampling in the 2-D frequency domain.
14. Let the 1-D finite support signal x be given over its support region $[0, N - 1]$ as

$$x(n) = 10 + 8 \cos \left[\frac{\pi}{2N} (2n + 1) \right] + 2 \cos \left[\frac{10\pi}{2N} (2n + 1) \right].$$

- (a) Sketch the plot of $x(n)$.
- (b) Find the DFT of x .
- (c) Find the DCT of x .
- (d) Use MATLAB to perform a 2-D version of this problem for the 2-D finite-support function x given over its support region $[0, N - 1]^2$ as

$$\begin{aligned} x(n_1, n_2) = & 10 + 8 \cos \left[\frac{\pi}{2N} (2n_1 + 1) \right] \cos \left[\frac{\pi}{2N} (2n_2 + 1) \right] \\ & + 2 \cos \left[\frac{10\pi}{2N} (2n_1 + 1) \right] \cos \left[\frac{10\pi}{2N} (2n_2 + 1) \right]. \end{aligned}$$

Please provide plots of $x(n_1, n_2)$, $|X(k_1, k_2)|$, and $X_C(k_1, k_2)$.

15. This problem concerns DCT properties, both in 1-D and 2-D cases.
- (a) Let $X_C(k)$ be the DCT of $x(n)$ with support $[0, N - 1]$. Find a simple expression for the DCT of $x(N - 1 - n)$. Express your answer directly in terms of $X_C(k)$.
- (b) Repeat for a 2-D finite support sequence $x(n_1, n_2)$ with support on $[0, N - 1]^2$, and find a simple expression for the 2-D DCT of $x(N - 1 - n_1, N - 1 - n_2)$ in terms of $X_C(k_1, k_2)$. (Please do not slight this part. It is the general case and, therefore, does not follow directly from the result of part a.)
16. This problem concerns filtering with the DCT. Take the version of DCT defined in the text with N points and consider only the 1-D case.
- (a) If possible, define a type of *symmetrical convolution* so that zero-phase filtering may be done with the DCT. Justify your definition of symmetrical convolution.

- (b) Consider a symmetrical zero-phase FIR filter with support $[-M, +M]$ and determine whether *linear convolution* can be obtained for a certain relation between M and N .
- (c) Is there a version of *sectioned convolution* that can be done here? Explain.
17. Let the signal (image) $x(n_1, n_2)$ have finite support $[0, M_1 - 1] \times [0, M_2 - 1]$. Then we have seen it is sufficient to sample its Fourier transform $X(\omega_1, \omega_2)$ on a uniform Cartesian grid of $N_1 \times N_2$ points, where $N_1 \geq M_1$ and $N_2 \geq M_2$. Here, we consider two such DFTs (i.e., $DFT_{N_1 \times N_2}$ and $DFT_{M_1 \times M_2}$).
- (a) Express $DFT_{N_1 \times N_2}$ in terms of $DFT_{M_1 \times M_2}$ as best as you can.
- (b) Consider the case where $N_i = L_i M_i$ for $i = 1, 2$, where the L_i are integers, and find an amplitude and phase closed-form representation for the resulting interpolator function.
18. We have been given an FIR filter h with support on the square $[0, N - 1] \times [0, N - 1]$. We must use a 512×512 -point row-column FFT to approximately implement this filter for a 512×512 -pixel image. Assume $N \ll 512$.
- (a) First, evaluate the number of complex multiplies used to implement the filter as a function of M for an $M \times M$ image, assuming the component 1-D FFT uses the Cooley-Tukey approach (assume M is a power of 2). Then specialize your result to $M = 512$. Assume that the FIR filter is provided as H in the DFT domain.
- (b) What portion of the output will be the correct (*linear convolution*) result?
19. We want to perform linear filtering of the signal (image) $x(n_1, n_2)$ with the FIR linear filter with impulse response coefficients $h(n_1, n_2)$ in the DFT domain. The supports of the image and filter impulse response are given as follows:
- $$\text{supp}\{x\} = [0, N - 1] \times [0, N - 1] \quad \text{and} \quad \text{supp}\{h\} = [-L, +L] \times [-L, +L].$$
- Assume that the positive integers L and N satisfy $N \gg L$. (Disregard any image positivity constraints.)
- (a) Employ DFTs of size $N \times N$ and specify in detail a method of placing the impulse response coefficients $h(n_1, n_2)$ into the DFT domain $[0, N - 1] \times [0, N - 1]$ so that the filter output $y = x * h$ will be represented by the product of the DFT coefficients $Y(k_1, k_2) = X(k_1, k_2)H(k_1, k_2)$, possibly with some delay (shift) and some circularity effect around the edges. Specify the location of these *bad* (due to circularity effect) pixels. (Note that you cannot take the DFT of h directly because of its symmetric support.)
- (b) Specify a mapping of the impulse response coefficients $h(n_1, n_2)$ onto the DFT domain $[0, N - 1] \times [0, N - 1]$ such that there is zero shift (delay). Justify your answer.
20. This problem concerns *2-D vectors*, which are rectangular $N_1 \times N_2$ arrays of values that correspond to finite sets of 2-D data, ordinarily thought of as matrices. We call them 2-D vectors because we want to process them by linear operators to produce output 2-D vectors. Such operators can be thought of as 4-D arrays,

the first two indices corresponding to a point in the output 2-D vector, and the last two indices corresponding to a point in the input 2-D vector. We will call these linear operators *4-D matrices*; thus, notationally we can write

$$\mathbf{Y} = \mathcal{H}\mathbf{X},$$

where bold capital values \mathbf{X}, \mathbf{Y} denote 2-D vectors and script value \mathcal{H} denotes the 4-D matrix.

- (a) Show that the set of 2-D $N_1 \times N_2$ vectors constitutes a finite dimensional vector space.
 - (b) Define the appropriate 2-D vector addition and 4-D matrix multiplication in this vector space.
 - (c) Show that this notationally simple device is equivalent to *stacking*, wherein the 2-D vectors \mathbf{X}, \mathbf{Y} are scanned in some way into 1-D vectors \mathbf{x}, \mathbf{y} and then the general linear operator becomes a 1-D or ordinary matrix \mathbf{H} , which has a block structure.⁸ Finally, relate the blocks in \mathbf{H} in terms of the elements of \mathcal{H} .
 - (d) Define a transpose for 2-D vectors \mathbf{X} and show it is not the same as regular matrix transpose. (Use superscript t for the 2-D vector transpose to distinguish it from regular matrix transpose that is denoted by superscript T .)
 - (e) How can we find the determinant $\det \mathcal{H}$? Then, if $\det \mathcal{H} \neq 0$, how can we define and then find \mathcal{H}^{-1} ?
21. Show that the 1-D Haar filters in [Example 4.4–2](#) satisfy the perfect reconstruction condition ([4.4–11](#)) and that the corresponding ISWT actually equals $x(n)$ by direct calculation in the 1-D frequency or time domain.

REFERENCES

- [1] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [2] A. V. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
- [3] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, p. 153–154 and 163–164, 1989.
- [4] H. Stark and J. W. Woods, *Probability and Random Processes with Application in Signal Processing*, 3rd Ed., Prentice-Hall, Upper Saddle River, NJ, 2002.
- [5] A. Croisier, D. Esteban, and C. Galand, “Perfect Channel Splitting by Use of Interpolation, Decimation, and Tree Decomposition Techniques,” *Proc. Intl. Conf. on Inform. Sciences/Systems*, Patras, Greece, August, pp. 443–446, 1976.

⁸Often the scanning used is *lexicographical order*, meaning left-to-right, and then top-to-bottom, also called *raster scan*.

- [6] D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," *Proc. Intl. Conf. Accoust., Speech, and Signal Proc. (ICASSP)*, May, pp. 191–195, 1977.
- [7] E. P. Simoncelli and E. H. Adelson, "Non-separable Extensions of Quadrature Mirror Filters to Multiple Dimensions," *Proc. IEEE*, vol. 78, No. 4, pp. 652–664, April 1990.
- [8] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 11, pp. 674–693, July 1989.
- [9] M. Vetterli and C. Herley, "Wavelets and Filter Banks: Theory and Design," *IEEE Trans. Signal Process.*, vol. 40, pp. 2207–2232, September 1992.
- [10] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT Algorithms with Reduced Number of Operations," *Signal Processing*, vol. 6, pp. 267–278, 1984.
- [11] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Two-Dimensional Filter Design

This chapter discusses the design of spatial and other 2-D digital filters. We will introduce both finite impulse response (FIR) and infinite impulse response (IIR) filter design. In the FIR case, we first present window function design of rectangular support filters. We then briefly present a 1-D to 2-D transformation method due to McClellan and a method based on successive projections onto convex sets of frequency domain specifications. In the IIR case, we look at computer-aided design methods that guarantee BIBO stability and also offer approximation for both magnitude and phase. We consider both conventional spatial-recursive filters as well as a more general class called fully recursive. Finally, we discuss subband/wavelet filter designs for use in the SWT.

5.1 FIR FILTER DESIGN

FIR support filters are often preferred in applications because of their ease of implementation and freedom from instability worries. However, they generally call for a greater number of multiplies and adds per output point. Also they typically require more temporary storage than IIR designs that are covered in [Section 5.2](#).

FIR Window Function Design

The method of window function design carries over simply from the 1-D case. We start with an ideal, but infinite order, impulse response $h_I(n_1, n_2)$, which may have been obtained by inverse Fourier transform of an ideal frequency response $H_I(\omega_1, \omega_2)$. We then proceed to generate our FIR filter impulse response $h(n_1, n_2)$ by simply multiplying the ideal impulse response by a prescribed *window function* $w(n_1, n_2)$ that has finite rectangular support:

$$h(n_1, n_2) \triangleq w(n_1, n_2)h_I(n_1, n_2),$$

with corresponding multiplication in the frequency domain,

$$H(\omega_1, \omega_2) = W(\omega_1, \omega_2) \otimes H_I(\omega_1, \omega_2),$$

where \circledast indicates periodic continuous-parameter convolution with fundamental period $[-\pi, +\pi] \times [-\pi, +\pi]$, specifically

$$H(\omega_1, \omega_2) = \frac{1}{(2\pi)^2} \int_{[-\pi, +\pi] \times [-\pi, +\pi]} W(\phi_1, \phi_2) H_I(\omega_1 - \phi_1, \omega_2 - \phi_2) d\phi_1 d\phi_2.$$

In applying this method, it is important that the window function support $\text{supp}(w)$ coincide with the largest coefficients of the ideal impulse response h_I , and such can be accomplished by shifting one or the other until they align.

We would like the window function and its Fourier transform to have the following properties:

- w should have rectangular $N_1 \times N_2$ support.
- w should approximate a circularly symmetric function (about its center point) and be real valued.
- The volume of w should be concentrated in the space domain.
- The volume of W should be concentrated in the frequency domain.

Ideally we would like W to be an impulse in frequency, but we know even from 1-D signal processing that this will not be possible for a space-limited function w . The desired symmetry in the window w will permit the window-designed impulse response h to inherit any symmetries present in the ideal impulse response h_I , which can be useful for implementation. In the 2-D case, we can consider two classes of window functions: *separable windows* and *circular windows*.

Separable (Rectangular) Windows

Here, we simply define the 2-D window function $w(n_1, n_2)$ as the product of two 1-D window functions,

$$w_s(n_1, n_2) \triangleq w_1(n_1)w_2(n_2).$$

This approach generally works well when the component functions are good 1-D windows.

Circular (Rotated) Windows

The circular window is defined in terms of a 1-D continuous time window function $w(t)$, as

$$w_c(n_1, n_2) \triangleq w\left(\sqrt{n_1^2 + n_2^2}\right),$$

which has the potential, at least, of offering better circular symmetry in the smoothing of the designed filter.

Common 1-D Continuous Time Windows

We define the following windows as centered on $t = 0$; however, you should keep in mind that when they are applied in filter design, they must be shifted to line up with the significant coefficients in the ideal impulse response.

- *Rectangular window:*

$$w(t) \triangleq \begin{cases} 1, & |t| < T, \\ 0, & \text{else.} \end{cases}$$

- *Bartlett (triangular) window:*

$$w(t) \triangleq \begin{cases} 1 - t/T, & 0 \leq t \leq T, \\ 1 + t/T, & -T \leq t \leq 0, \\ 0, & \text{else.} \end{cases}$$

- *Hanning window:*

$$w(t) \triangleq \begin{cases} \frac{1}{2}(1 + \cos \pi t/T), & |t| < T, \\ 0, & \text{else.} \end{cases}$$

- *Kaiser window:*

$$w(t) \triangleq \begin{cases} I_0\left(\beta\sqrt{1 - (t/T)^2}\right)/I_0(\beta), & |t| < T, \\ 0, & \text{else,} \end{cases}$$

where $I_0(t)$ is the modified Bessel function of the first kind and of zero order [1]. The free parameter β in this definition means that the Kaiser window is actually a family of windows. As β is varied over a range, typically $[0, 8]$, the transition bandwidth goes up while the filter attenuation in the stop band goes down, in such a manner as to better or equal the performance of the other known 1-D window functions. We thus concentrate on only Kaiser window function designs for our 2-D filters.

Approximate design formulas have been developed for 2-D circular symmetric lowpass filters [2] to estimate the required values of the filter size N_1 and N_2 and the Kaiser parameter¹ β . They are based on the desired filter transition bandwidth $\Delta\omega$ and stopband attenuation ATT , defined as follows:

$$\Delta\omega \triangleq \omega_s - \omega_p,$$

where ω_s and ω_p denote the circular radii of the desired stopband and passband, respectively, and each with maximum value of π . The attenuation parameter ATT is given as

$$ATT \triangleq -20\log_{10}(\sqrt{\delta_s\delta_p}),$$

where δ_s and δ_p denote the desired peak frequency domain errors in the stopband and passband, respectively. The estimated filter orders are given for the square support case $N_1 = N_2$ as *separable*:

$$N_s \approx \frac{ATT - 8}{2.10 \Delta\omega},$$

¹This β parameter is denoted as α in Oppenheim, Schaffer, and Buck [3].

and *circular*:

$$N_c \approx \frac{ATT - 7}{2.18 \Delta \omega}.$$

The estimate of β is then given as *separable*:

$$\beta_s \approx \begin{cases} 0.42(ATT - 19.3)^{0.4} + 0.089(ATT - 19.3), & 20 < ATT < 60, \\ 0, & \text{else,} \end{cases}$$

and *circular*:

$$\beta_c \approx \begin{cases} 0.56(ATT - 20.2)^{0.4} + 0.083(ATT - 20.2), & 20 < ATT < 60, \\ 0, & \text{else.} \end{cases}$$

Example 5.1–1: Window Design Using MATLAB

We have designed an 11×11 lowpass filter with filter cutoff $f_c = 0.3$ ($\omega_c = 0.6\pi$) and using two Kaiser windows, with $\beta = 8$, both separable and circular. Figure 5.1–1 shows the magnitude response of the filter obtained using the separable Kaiser window. The contour plot of the magnitude data is shown in Figure 5.1–2. Figure 5.1–3 shows the 11×11 impulse response of the designed FIR filter. Figure 5.1–4 shows a contour plot of the impulse response to demonstrate its degree of “circularity.”

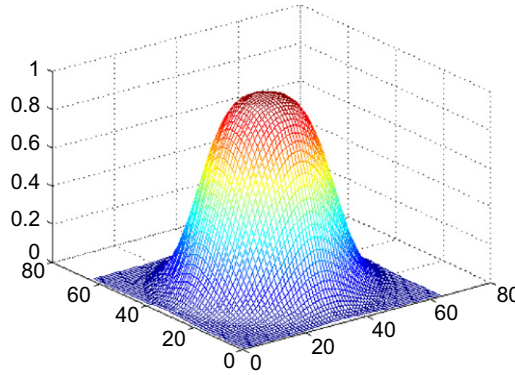
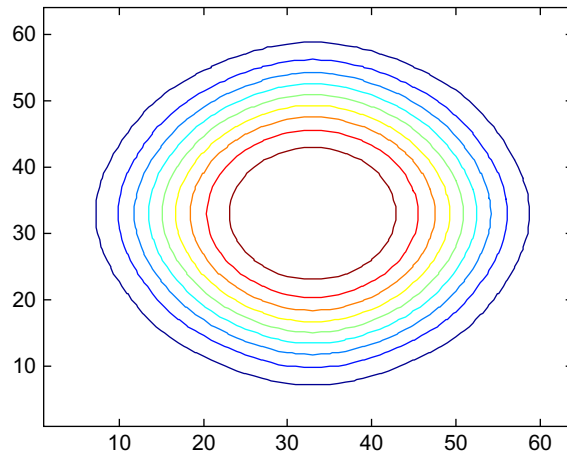


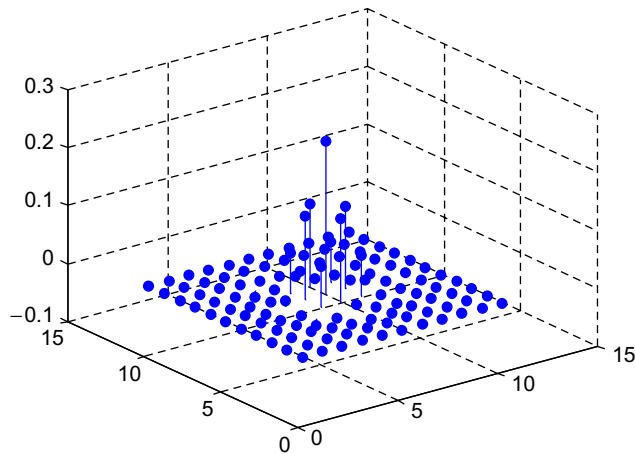
FIGURE 5.1–1

Magnitude response of Kaiser window FIR filter design with $\beta = 8$ (64×64 DFT with (0,0) shifted to center).

The next set of figures shows corresponding results for a circular Kaiser window. Figure 5.1–5 shows the magnitude response, followed by Figure 5.1–6 showing its contour plot in frequency. Figure 5.1–7 shows the corresponding impulse response plot, and then Figure 5.1–8 shows the contour plot. Note that both the circular- and separable-designed

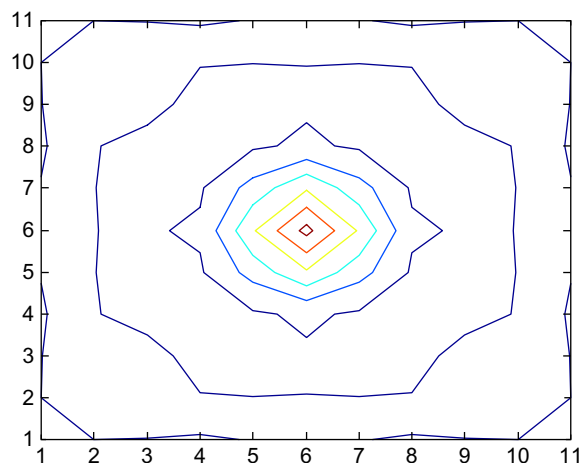
**FIGURE 5.1–2**

Contour plot of separable Kaiser window-designed filter $\beta = 8$ (64×64 DFT with (0,0) shifted to center).

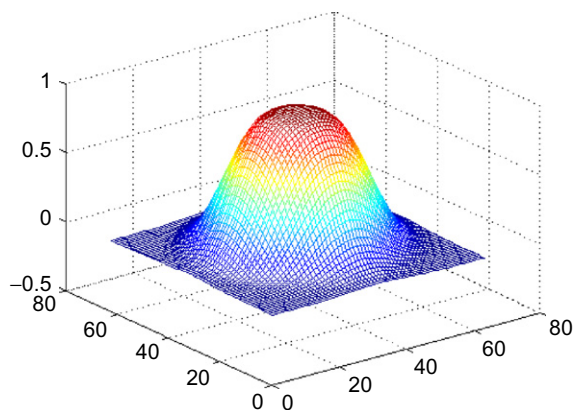
**FIGURE 5.1–3**

Plot of 11×11 impulse response.

filters display a lot of circular symmetry, which they inherit from the exact circular symmetry of the ideal response H_I . Both designs are comparable. A lower value of β would result in lower transition bandwidth, but less attenuation in the stopband and more ripple in the passband. ■

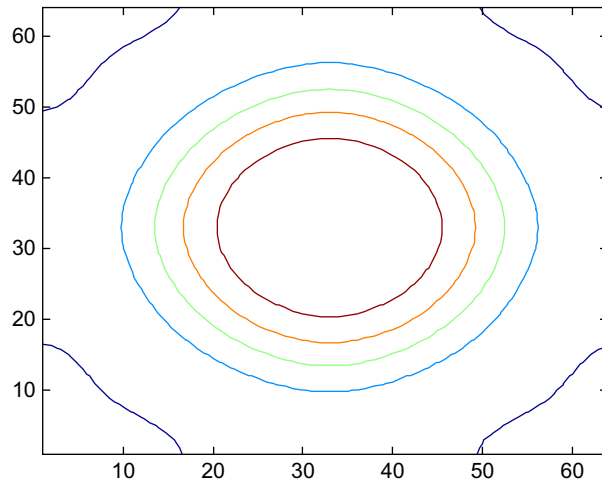
**FIGURE 5.1–4**

Contour plot of impulse response.

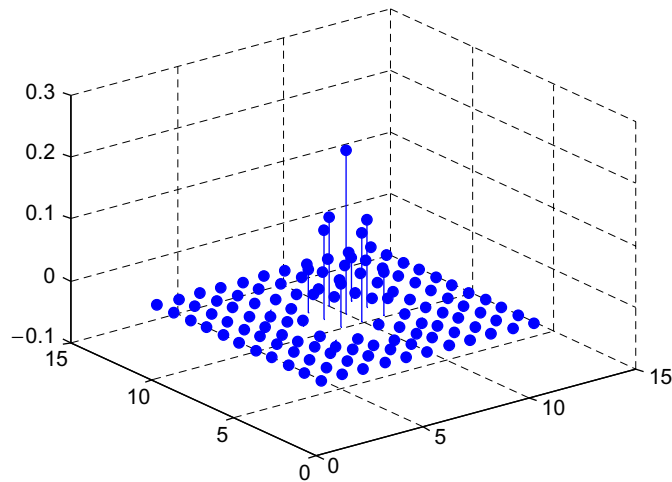
**FIGURE 5.1–5**

$\beta = 8$, but with circular Kaiser window (64×64 DFT with (0,0) shifted to center).

The MATLAB .m files WinDes permit easy experimentation with Kaiser window design of spatial FIR filters. The program WinDesS.m uses separable window design, while WinDesC.m uses the circular window method. Both programs make use of the MATLAB image processing toolbox function fwind1.m, which uses 1-D windows. The other toolbox function fwind2.m does window design with a 2-D window that you supply. It is not used in either WinDes program. As an exercise, try using WinDesC with the parameters of [Example 5.1–1](#), except that $\beta = 2$. These programs are available at the book's Web site.

**FIGURE 5.1–6**

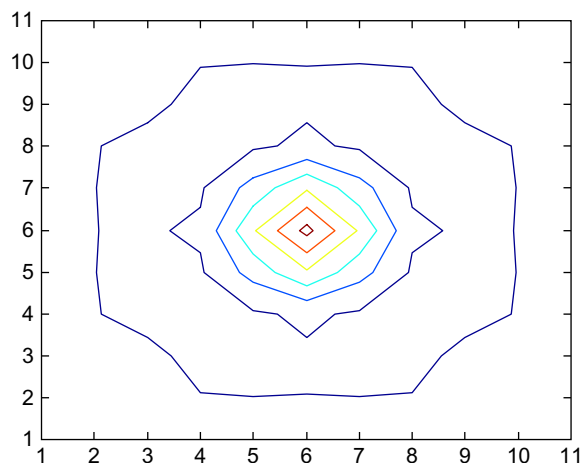
Contour plot of magnitude response (64×64 DFT with (0,0) shifted to center).

**FIGURE 5.1–7**

11×11 impulse response of Kaiser circular window-designed filter.

Example 5.1–2: Filter Eric Image

We take the filter designed with the circular window and apply it to filtering the Eric image from Figure 1.1–7 of Chapter 1. The result is shown in Figure 5.1–9 where we note the visually blurring effect of the lowpass filter.

**FIGURE 5.1–8**

Contour plot of impulse response.

**FIGURE 5.1–9**

Lowpass filtering of the Eric image from Figure 1.1–7 of Chapter 1.

We now turn to an FIR design technique that transforms 1-D filters to obtain 2-D filters.

Design by Transformation of 1-D Filter

Another common and powerful design method is by transformation. The basic idea starts with a 1-D filter and then transforms it to two dimensions in such a way that the 2-D filter response is constant along contours specified by the transformation. Thus, for contours with an approximate circular shape, and a 1-D lowpass filter, we can achieve approximate circular symmetry in a 2-D lowpass filter. The basic idea is due to McClellan [4], and it is developed much more fully in Lim's text [5] than

here, where we present only the basics. The method is also available in the MATLAB image processing toolbox.

We start with a 1-D FIR filter, of type I with zero phase [3], so that we can write the frequency response as the real function

$$H(\omega) = \sum_{n=0}^M a(n) \cos(n\omega), \quad (5.1-1)$$

where the filter coefficients $h(n)$ can be expressed in terms of the $a(n)$. Next, we rewrite this equation in terms of Chebyshev polynomials $\cos(n\omega) = T_n[\cos \omega]$ as

$$H(\omega) = \sum_{n=0}^M a'(n) \cos^n \omega, \quad (5.1-2)$$

where the $a'(n)$ can be expressed in terms of the $a(n)$ (see end-of-chapter problem 8). Note that the first few Chebyshev polynomials are given as

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1,$$

and in general

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n > 1.$$

Next, we introduce a transformation from the 2-D frequency plane to $\cos \omega$, given as [4]

$$F(\omega_1, \omega_2) = A + B \cos \omega_1 + C \cos \omega_2 + D \cos(\omega_1 - \omega_2) + E \cos(\omega_1 + \omega_2), \quad (5.1-3)$$

where the constants A, B, C, D , and E are adjusted to the constraint $|F(\omega_1, \omega_2)| \leq 1$ so that F will equal $\cos \omega$ for some ω .

At this point, we parenthetically observe that when the 1-D filter is expressed in terms of a Z-transform, the term $\cos \omega$ will map over to $\frac{1}{2}(z + z^{-1})$, and the transformation F will map into the 2-D Z-transform of the 1×1 order symmetric FIR filter f given as

$$f(n_1, n_2) = \begin{Bmatrix} \frac{1}{2}D & \frac{1}{2}C & \frac{1}{2}E \\ \frac{1}{2}B & A & \frac{1}{2}B \\ \frac{1}{2}E & \frac{1}{2}C & \frac{1}{2}D \end{Bmatrix},$$

with axes $\rightarrow n_1$ and n_2 upwards, with $\mathbf{0}$ and the center. Thus we can take a realization of the 1-D filter and, replacing the delay boxes z^{-1} with the filter $F(z_1, z_2)$, obtain a realization of the 2-D filter. Of course, this can be generalized in various ways, including the use of higher order transformations and the other types (II and IV) of 1-D FIR filters.

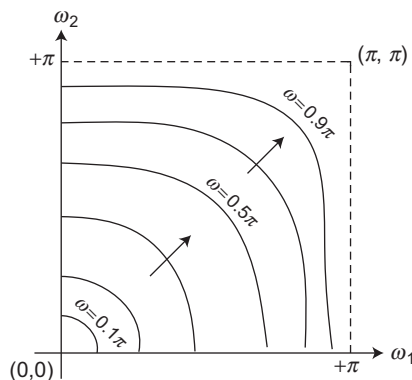
**FIGURE 5.1-10**

Illustration of 1×1 order McClellan transformation for nearly circular symmetric contours.

If we take the values $A = -0.5$, $B = C = 0.5$, and $D = E = 0.25$, we get contours $F(\omega_1, \omega_2)$ with a nearly circular shape, even near $(\pm\pi, \pm\pi)$, as is sketched in Figure 5.1-10, which is very nearly circular in shape out to about midfrequency $\approx \pi/2$.

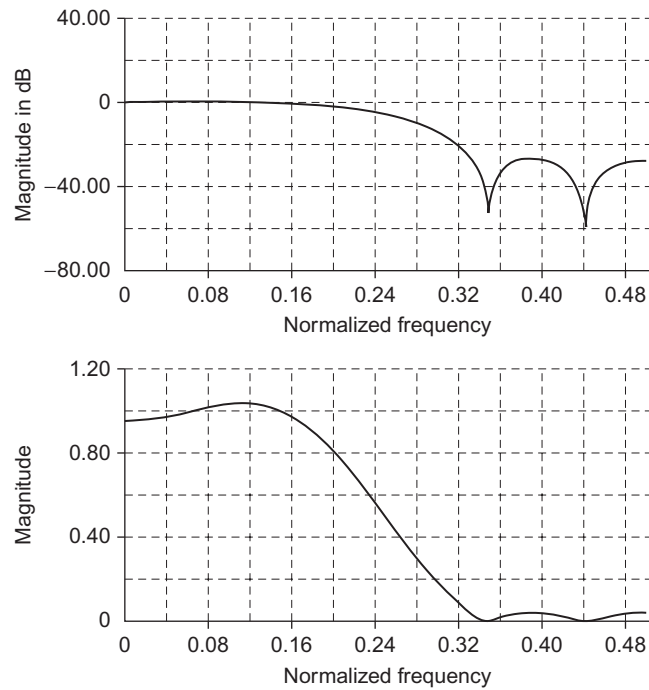
If we start with a 1-D filter that is optimal in the Chebyshev sense (i.e., l_∞ optimal in the magnitude response), then it has been shown [5] that the preceding 1×1 order transformation preserves optimality in this sense for the 2-D filter *if* the passband and stopband contours of the desired filter are isopotentials of the transformation.

The following examples show the transformation of 1-D optimal Chebyshev lowpass filters into 2-D filters via this method.

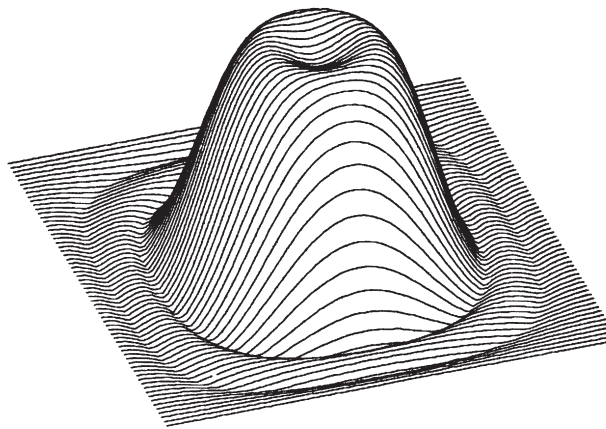
Example 5.1-3: 9×9 Lowpass Filter

We start with the design of a nine-tap (eighth-order) FIR lowpass filter with passband edge $f_p = 0.166$ ($\omega_p = 0.332\pi$) and stopband edge $f_s = 0.333$ ($\omega_s = 0.666\pi$), using the Remez (also called Parks-McClellan) algorithm. The result is a passband ripple of $\delta = 0.05$ and a stopband attenuation $ATT = 30$ dB, and the resulting linear and logarithmic magnitude frequency response shown in Figure 5.1-11. When this 9-tap filter is transformed with the nearly circular 1×1 -order McClellan transform, we obtain the 2-D amplitude response shown in Figure 5.1-12, where the zero frequency point $\mathbf{f} = \mathbf{0}$ ($\boldsymbol{\omega} = \mathbf{0}$) is in the middle of this 3-D perspective plot.

We note that this design has resulted in a very good approximation of circular symmetry, as we expect from the contour plot of Figure 5.1-10 for the transformation used. For this particular transformation, at $\omega_2 = 0$, we get $\cos \omega = \cos \omega_1$, so that the passband and stopband edges are located at the same frequencies, on the axes at least, as those of the 1-D prototype filter (i.e., $f_p = 0.166$ ($\omega_p = 0.332\pi$) and $f_s = 0.333$ ($\omega_s = 0.666\pi$), respectively).

**FIGURE 5.1–11**

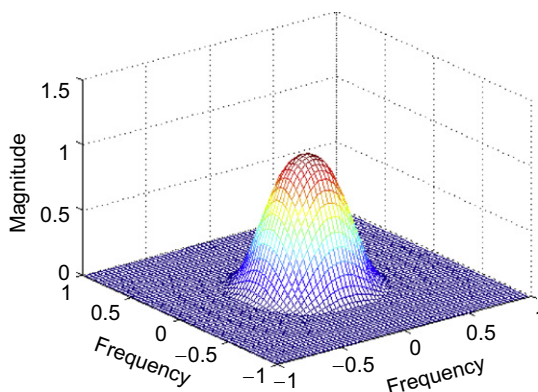
Magnitude response of nine-tap FIR type I filter.

**FIGURE 5.1–12**

Plot of a 9×9 FIR filter designed as a 2-D transform of a nine-tap lowpass filter.

Example 5.1–4: Application of Transform Lowpass Filter

In this example, we design and apply a transformation-designed lowpass filter to a digital image by use of MATLAB functions. Using the transformation of [Example 5.1–3](#), we convert a lowpass 1-D filter with passband edge $\omega_p = 0.025\pi$ and stopband edge $\omega_p = 0.25\pi$ into an 11×11 near-circular symmetric lowpass filter. The resulting 2-D frequency response is shown in [Figure 5.1–13](#). The input image is shown in [Figure 5.1–14](#), and the output image is shown in [Figure 5.1–15](#). The difference image (biased up by +128 for display on $[0, 255]$) is shown in [Figure 5.1–16](#).

**FIGURE 5.1–13**

A lowpass filter for image filtering. Note: MATLAB defines frequency $f = \omega/\pi$.

**FIGURE 5.1–14**

Lena 512×512 input image.

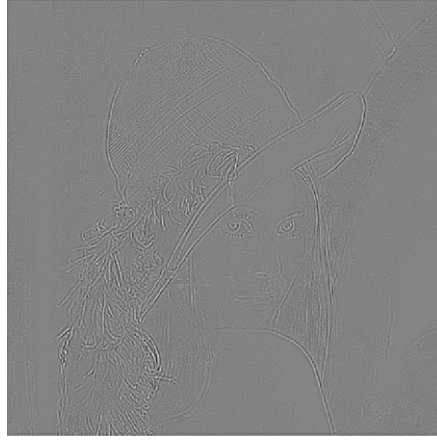
**FIGURE 5.1-15**

Output of lowpass filter in [Example 5.1-4](#).

**FIGURE 5.1-16**

Corresponding difference image (biased for display by +128).

The corresponding MATLAB .m file `MacDesFilt.m` is given at the book's Web site. There you can experiment with changing the passband, filter type, image, etc. (Note this routine requires MATLAB with the image processing toolbox.) Similarly, using an approximately circular highpass filter, we can get the output shown in [Figure 5.1-17](#), where we note the similarity to [Figure 5.1-16](#).

**FIGURE 5.1–17**

Output of McClellan transformed near circular highpass filter.

Projection Onto Convex Sets

The method of *projection onto convex sets* (POCS) is quite general and has been applied to several image processing optimization problems [6]. Here, we follow an application to 2-D FIR filter design due to Abo-Taleb and Fahmy [7]. Using an assumed symmetry in the ideal function and the desired FIR filter, we rewrite the filter response in terms of a set of frequency domain basis functions as

$$H(\omega_1, \omega_2) = \sum_{k=1}^M a(k) \phi_k(\omega_1, \omega_2),$$

where the filter coefficients $h(n_1, n_2)$ can be simply expressed in terms of the $a(k)$, using built-in symmetries assumed for h . Next, we densely and uniformly discretize the basic frequency cell with N^2 points as $\mathbf{x}_i = (\omega_1^i, \omega_2^i)$, for $i = 1, \dots, N^2$. Then we can express the filter response at grid point \mathbf{x}_i as

$$H(\mathbf{x}_i) = \langle \mathbf{a}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle,$$

where \mathbf{a} is a column vector of the $a(k)$ and $\boldsymbol{\phi}$ is a vector of the ϕ_k , each of dimension M , and $\langle \cdot, \cdot \rangle$ denotes the conventional inner product. The frequency domain error at the point \mathbf{x}_i then becomes

$$e(\mathbf{x}_i) = I(\mathbf{x}_i) - \langle \mathbf{a}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle.$$

We can then write the approximation problem as

$$|I(\mathbf{x}_i) - \langle \mathbf{a}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle| \leq \delta \quad \text{for all } i = 1, \dots, N^2.$$

To apply the POCS method, we must first observe that the sets

$$Q_i \triangleq \{\mathbf{a} : |I(\mathbf{x}_i) - \langle \mathbf{a}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle| \leq \delta\}$$

are *convex*² and closed. We note that we want to find a vector \mathbf{a} that is contained in the set intersection

$$Q \triangleq \cap_{i=1}^{N^2} Q_i,$$

if such a point exists. In that case, the POCS algorithm is guaranteed to converge to such a point(s) [7]. Note that the existence of a solution will depend on the tolerance δ , and that we do not know how small the tolerance should be. What this means in practice is that a sequence of problems will have to be solved for a decreasing set of δ 's until the algorithm fails to converge, at which time the last viable solution is taken as the result. Of course, one could also specify a needed value δ and then increase the filter order, in terms of M , until a solution is obtained. The basic POCS algorithm can be given as follows.

Basic POCS Algorithm

1. Scan through the frequency grid points \mathbf{x}_i to find the frequency \mathbf{x}_p of maximum error $e(\mathbf{x}_p) = e_p$.
2. If $e_p \leq \delta$, stop.
3. Else, update the coefficient vector as the orthogonal projection onto Q_p [6, 7],

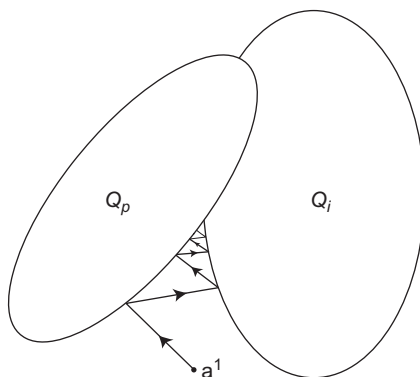
$$\mathbf{a}^{n+1} = \begin{cases} \mathbf{a}^n - (\delta - |e_p|)\boldsymbol{\phi}(\mathbf{x}_p) \frac{\text{sgn}(e_p)}{\|\boldsymbol{\phi}(\mathbf{x}_p)\|}, & \text{if } e_p > \delta, \\ \mathbf{a}^n, & \text{else.} \end{cases}$$

4. Return to step 1.

The key step of this algorithm is step 3, which performs an orthogonal projection of the M -dimensional filter coefficient vector \mathbf{a}^n onto the constraint set Q_p . As the algorithm progresses, the solution point \mathbf{a} may move into and out of any given set many times, but is guaranteed to converge finally to a point in the intersection of all the constraint sets, *if such a point exists*. An illustrative diagram of POCS convergence is shown in Figure 5.1–18.

The references [6, 7] contain several examples of filters designed by this method. The obtained filters appear to be quite close to the Chebyshev optimal linear phase FIR designs of Harris and Mersereau [8], which are not covered here. The POCS designs appear to have a significant computational advantage, particularly for larger filter supports.

²A convex set Q is one for which, if points \mathbf{p}_1 and \mathbf{p}_2 are in the set, then so must be $\mathbf{p} = \alpha\mathbf{p}_1 + (1 - \alpha)\mathbf{p}_2$ for all $0 \leq \alpha \leq 1$.

**FIGURE 5.1-18**

An illustration of the convergence based on orthogonal projection onto two convex sets.

5.2 IIR FILTER DESIGN

In this section, we first look at conventional 2-D recursive filter design. This is followed by a discussion of a generalization called fully recursive filters, which offers the potential of even better performance.

2-D Recursive Filter Design

The Z-transform system function of a spatial IIR or recursive filter is given as

$$H(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)},$$

where both B and A are polynomial³ functions of z_1, z_2 , which in the spatial domain yields the computational procedure, assuming $a_{0,0} = 1$,

$$\begin{aligned} y(n_1, n_2) = & - \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2} y(n_1 - k_1, n_2 - k_2) \\ & + \sum_{(k_1, k_2) \in \mathcal{R}_b} b_{k_1, k_2} x(n_1 - k_1, n_2 - k_2), \end{aligned}$$

where the coefficient support regions of the denominator and numerator are denoted as \mathcal{R}_a and \mathcal{R}_b , respectively. Here, we consider a so-called *direct form* or unfactored design, but factored designs are possible, and even preferable for implementation in finite word-length arithmetic, as would be expected from the 1-D case. However, as

³Strictly speaking, the “polynomial” may include both positive and negative powers of the z_i , but we do not make the distinction here. We call both polynomials, if they are just finite order.

we have seen earlier, various factored forms will not be equivalent due to the lack of a finiteorder factorization theorem in two and higher dimensions.

Typical Design Criteria

The choice of error criteria is important and related to the expected use of the filter. In addition to the choice of either spatial-domain or frequency-domain error criteria, or a combination of both, there is the choice of error norm. The following error criteria are often used:

- *Space-domain design:* Often a least-squares design criteria is chosen,

$$\|h_I(n_1, n_2) - h(n_1, n_2)\|_2,$$

via use of the so-called l^2 error norm

$$\|f\|_2 \triangleq \sqrt{\sum \sum |f|^2(n_1, n_2)}.$$

- *Magnitude-only approximation:*

$$\| |H_I|(\omega_1, \omega_2) - |H|(\omega_1, \omega_2) \|,$$

where $|H_I|$ denotes an ideal magnitude function and $|H|$ is the magnitude of the filter being designed. The most common choice for the norm $\|\cdot\|$ is the so-called L^2 norm

$$\|F\|_2 \triangleq \sqrt{\int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} |F|^2(\omega_1, \omega_2) d\omega_1 d\omega_2}.$$

Magnitude-only design is generally a carryover from 1-D filter design where the common IIR design methods are based on transforming an analog filter via the bilinear transform. These methods lead to very good filter magnitude response, but with little control over the phase. For single channel audio systems, these filter phase distortions can be unimportant. For image processing, though, where 2-D filters see a big application, phase is very important. So a magnitude-only design of an image processing filter probably will not be adequate.

- *Zero-phase design:*

$$\| H_I(\omega_1, \omega_2) - |H|^2(\omega_1, \omega_2) \|,$$

where the ideal function H_I is assumed to be *real valued and nonnegative*, often the case in image processing. The filter is then realized by two passes. In the first pass, the image data is filtered by $h(n_1, n_2)$ in its stable direction, generally from left-to-right and then top-to-bottom, the so-called *raster scan* of image processing. The output data from this step is then filtered by $h^*(-n_1, -n_2)$ in its stable direction, from right-to-left and bottom-to-top. The overall realized frequency response is then $|H|^2(\omega_1, \omega_2)$, which is guaranteed zero-phase. Note that this method only yields non-negative frequency responses.

- *Magnitude and phase (delay) design:*

$$\|H_I(\omega_1, \omega_2) - H(\omega_1, \omega_2)\|$$

or

$$\||H_I|(\omega_1, \omega_2) - |H|(\omega_1, \omega_2)\| + \lambda \|\arg H_I(\omega_1, \omega_2) - \arg H(\omega_1, \omega_2)\|,$$

where the Lagrange parameter λ controls the weight given to phase error versus magnitude error. A particular choice of λ will give a prescribed amount of phase error. This critical value of λ , though, will usually only be approximately determined after a series of designs for a range of λ values.

Space-Domain Design

We look at two methods, first the design method called Padé approximation, which gives exact impulse response values but over a usually small range of points. Then we look at least-squares extensions, which minimize a modified error.

Padé Approximation

This simple method carries over from the 1-D case. Let

$$e(n_1, n_2) = h_I(n_1, n_2) - h(n_1, n_2).$$

The squared l^2 norm then becomes

$$\|e\|_2^2 = \sum e^2(n_1, n_2) \\ \triangleq f(\mathbf{a}, \mathbf{b}),$$

a nonlinear function of the denominator and numerator coefficient vectors \mathbf{a} and \mathbf{b} , respectively. The Padé approximation method gives a linear closed-form solution that achieves zero-impulse response error over a certain support region, whose number of points equals $\dim(\mathbf{a}) + \dim(\mathbf{b})$.

Let $x = \delta$ so that $y = h$; then

$$h(n_1, n_2) = - \sum_{(k_1, k_2) > (0,0)} a(k_1, k_2) h(n_1 - k_1, n_2 - k_2) + b(n_1, n_2) \\ \neq h_I(n_1, n_2),$$

where we have assumed that coefficient $a(0,0) = 1$ without loss of generality. The error function then becomes

$$e(n_1, n_2) = h_I(n_1, n_2) + \sum_{(k_1, k_2) > (0,0)} a(k_1, k_2) h(n_1 - k_1, n_2 - k_2) + b(n_1, n_2),$$

which is nonlinear in the parameters a and b through the function h . To avoid the nonlinearities, we now define the *modified error*:

$$e_M(n_1, n_2) \triangleq h_I(n_1, n_2) + \sum_{(k_1, k_2) > (0,0)} a(k_1, k_2) h_I(n_1 - k_1, n_2 - k_2) - b(n_1, n_2) \\ = a(n_1, n_2) * h_I(n_1, n_2) - b(n_1, n_2). \quad (5.2-1)$$

Comparison of these two equations reveals that we have substituted h_I for h inside the convolution with a , which may be reasonable given a presumed smoothing effect from coefficient sequence a . To this extent we can hope that $e_M \approx e$. Note in particular, though, that the modified error e_M is linear in the coefficient vectors a and b .

If $\dim(a) = p$ and $\dim(b) = q + 1$, then there are $N = p + q + 1$ unknowns. We can then set $e_M = 0$ on $\mathcal{R}_{\text{Padé}} = \{\text{a connected } N \text{ point region}\}$. We have to choose the region so that the filter masks slide over the set, rather than bringing new points (pixels) into the equations. A simple example suffices to illustrate this linear FIR design method.

Example 5.2–1: Padé Approximation

Consider the 1×1 -order system function

$$H(z_1, z_2) = \frac{b_{00} + b_{01}z_2^{-1}}{1 + a_{10}z_1^{-1} + a_{01}z_2^{-1}},$$

where we have $p = 2$ and $q + 1 = 2$, equivalent to $N = 4$ unknowns. We look for a first quadrant support for the impulse response h and set $\mathcal{R}_{\text{Padé}} = \{(0, 0), (1, 0), (0, 1), (1, 1)\}$. Setting $e_M = 0$, we have the following recursion:

$$h_I(n_1, n_2) = -a_{10}h_I(n_1 - 1, n_2) - a_{01}h_I(n_1, n_2 - 1) + b_{00}\delta(n_1, n_2) + b_{01}\delta(n_1, n_2 - 1).$$

Assuming zero initial (boundary) conditions, we evaluate this equation on the given region $\mathcal{R}_{\text{Padé}}$ to obtain the four equations,

$$\begin{aligned} h_I(0, 0) &= b_{00}, \\ h_I(1, 0) &= -a_{10}h_I(0, 0), \\ h_I(0, 1) &= -a_{01}h_I(0, 0) + b_{01}, \\ h_I(1, 1) &= -a_{10}h_I(0, 1) - a_{01}h_I(1, 0), \end{aligned}$$

which can be easily solved for the four coefficient unknowns. ■

We have not yet mentioned that the filter resulting from Padé approximation may not be stable! Unfortunately, this is so. Still if it is stable, and if the region is taken large enough, good approximations can result. If the ideal impulse response is stable (i.e., $h_I \in l^1$), then the Padé approximate filter *should* also be stable, again if the region is taken large enough. We also note that, because of the shape of the Padé region $\mathcal{R}_{\text{Padé}}$, effectively honoring the output mask of the filter, that $e_M = e$ there, and so the obtained impulse response values are exact *on that region*. Thus, the key is to include in the region $\mathcal{R}_{\text{Padé}}$ all the “significant values” of the ideal impulse response. Of course, this is not always easy and can lead to rather large filter orders.

We next turn to an extension of Padé approximation called Prony's method, which minimizes the l^2 spatial domain modified error over (in theory) its entire support region, rather than just setting it to zero over the small region $\mathcal{R}_{\text{Padé}}$.

Prony's Method (Shank's)

This is a linear least-squares method, which minimizes the modified error (5.2-1), but again ignores stability. We write

$$\begin{aligned}\mathcal{E}_M &= \sum e_M^2(n_1, n_2) \\ &= \sum_{\mathcal{R}_b} e_M^2(n_1, n_2) + \sum_{\mathcal{R}_b^c} e_M^2(n_1, n_2) \\ &\triangleq \mathcal{E}_{M, \mathcal{R}_b} + \mathcal{E}_{M \mathcal{R}_b^c},\end{aligned}$$

making the obvious definitions, and note that the second error term will be independent of coefficient vector \mathbf{b} . This because on \mathcal{R}_b^c we can write

$$\begin{aligned}e_M(n_1, n_2) &= a(n_1, n_2) * h_I(n_1, n_2) \\ &= f(\mathbf{a}),\end{aligned}$$

so we can perform linear least squares to minimize $\mathcal{E}_{M \mathcal{R}_b^c}$ over \mathbf{a} . Again, we emphasize that we cannot just choose $\mathbf{a} = \mathbf{0}$ to make the modified error $e_M = 0$, since we have the constraint $a(0, 0) = 1$ such that $a(0, 0)$ is not an element in the design vector \mathbf{a} .

Considering the practical case where there are a finite number of points to consider, we can order the values $e_M(n_1, n_2)$ on \mathcal{R}_b^c onto a vector, say \mathbf{e}_M , and then minimize $\mathbf{e}_M^T \mathbf{e}_M$ over the choice of denominator coefficient vector \mathbf{a} . A typical term in $\mathbf{e}_M^T \mathbf{e}_M$ would look like

$$e_M^2(n_1, n_2) = [h_I(n_1, n_2) + a(1, 0)h_I(n_1 - 1, n_2) + a(0, 1)h_I(n_1, n_2 - 1) + \dots]^2$$

since $a(0, 0) = 1$. Thus taking partials with respect to the $a(k_1, k_2)$, we obtain

$$\begin{aligned}\partial(\mathbf{e}_M^T \mathbf{e}_M) / \partial a(k_1, k_2) &= 2 \sum_{n_1, n_2} h_I(n_1 - k_1, n_2 - k_2) \left[h_I(n_1, n_2) + a(1, 0)h_I(n_1 - 1, n_2) \right. \\ &\quad \left. + a(0, 1)h_I(n_1, n_2 - 1) + \dots \right] \\ &= 2 \sum_{n_1, n_2} h_I(n_1 - k_1, n_2 - k_2) \left[h_I(n_1, n_2) \right. \\ &\quad \left. + \sum_{l_1, l_2} a(l_1, l_2)h_I(n_1 - l_1, n_2 - l_2) \right],\end{aligned}$$

and setting these partial derivatives to zero, we obtain the so-called *normal equations*

$$\sum_{l_1, l_2} R_I(k_1 - l_1, k_2 - l_2) a(l_1, l_2) = -R_I(k_1, k_2) \text{ for } (k_1, k_2) \in \text{supp}\{a\}, \quad (5.2-2)$$

with the definition of the “correlation terms”

$$R_I(k_1, k_2) \triangleq \sum_{n_1, n_2} h_I(n_1 - k_1, n_2 - k_2) h_I(n_1, n_2).$$

Finally, (5.2-2) can be put into matrix-vector form and solved for the denominator coefficient vector \mathbf{a} in the case when the matrix corresponding to the “correlation function” is nonnegative definite, fortunately most of the time. Turning now to the first error term $\mathcal{E}_{M, \mathcal{R}_b}$, we can write

$$\begin{aligned} e_M(n_1, n_2) &= a(n_1, n_2) * h_I(n_1, n_2) - b(n_1, n_2) \\ &= 0, \end{aligned} \quad (5.2-3)$$

upon setting $b(n_1, n_2) = a(n_1, n_2) * h_I(n_1, n_2)$, using $a(0, 0) = 1$ and the values of \mathbf{a} obtained in the least-squares solution. We then obtain $\mathcal{E}_{M, \mathcal{R}_b} = 0$, and hence have achieved a minimum for the total modified error \mathcal{E}_M .

Example 5.2-2: 1 × 1-order Case

Consider the design problem with a 1×1 -order denominator with variable coefficients $\{a(1, 0), a(0, 1), a(1, 1)\}$ and a numerator consisting of coefficients $\{b(0, 0), b(1, 0)\}$. Then set $\mathcal{R}_b^c = \{n_1 \geq 2, n_2 = 0\} \cup \{n_1 \geq 0, n_2 > 0\}$ for a designed first quadrant support. The complementary region is then $\mathcal{R}_b = \{(0, 0), (1, 0)\}$. For a given ideal impulse response, with $\text{supp}\{h_I\} = \{n_1 \geq 0, n_2 \geq 0\}$, we next compute $R_I(k_1, k_2)$ for $(k_1, k_2) \in \mathcal{R}_b^c$. Then we can write the normal equations as

$$\begin{aligned} R_I(0, 0)a(1, 0) + R_I(1, -1)a(0, 1) + R_I(0, -1)a(1, 1) &= -R_I(1, 0), \\ R_I(-1, 1)a(1, 0) + R_I(0, 0)a(0, 1) + R_I(-1, 0)a(1, 1) &= -R_I(0, 1), \\ R_I(0, 1)a(1, 0) + R_I(1, 0)a(0, 1) + R_I(0, 0)a(1, 1) &= -R_I(1, 1), \end{aligned}$$

which can be put into matrix form

$$\mathbf{R}_I \mathbf{a} = -\mathbf{r}_I$$

and solved for vector $\mathbf{a} = [a(1, 0), a(0, 1), a(1, 1)]^T$. Finally, we go back to (5.2-3) and solve for $b(0, 0)$ and $b(1, 0)$ as $b(n_1, n_2) = a(n_1, n_2) * h_I(n_1, n_2)$ for $(n_1, n_2) = (0, 0)$ and $(1, 0)$. ■

As mentioned previously, there is no constraint of filter stability here. The resulting filter may be stable or it may not. Fortunately, the method has been found useful in practice, depending on the stability of the ideal impulse response h_I that is being approximated. More on Prony’s method, and an iterative improvement method, is contained in Lim [5].

Fully Recursive Filter Design

The Z-transform system function of a *fully recursive filter* (FRF) is given as

$$H(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)},$$

where both B and A are rational functions of z_1 and polynomial functions of z_2 . The 2-D difference equation is given by

$$\sum_{k_1=-\infty}^{+\infty} \sum_{k_2=0}^{L_D} a(k_1, k_2) y(n_1 - k_1, n_2 - k_2) = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=0}^{L_N} b(k_1, k_2) x(n_1 - k_1, n_2 - k_2),$$

which can be expressed in *row-operator* form as

$$a_0(n_1) * y_{n_2}(n_1) = - \sum_{k_2=1}^{L_D} a_{k_2}(k_1) * y_{n_2-k_2}(n_1) + \sum_{k_2=0}^{L_N} b_{k_2}(k_1) * x_{n_2-k_2}(n_1), \quad (5.2-4)$$

where the 1-D coefficient row sequences are given as

$$\begin{aligned} a_0(n_1) &\triangleq a(n_1, 0), & b_0(n_1) &\triangleq b(n_1, 0), \\ a_{n_2}(n_1) &\triangleq a(n_1, n_2), & b_{n_2}(n_1) &\triangleq b(n_1, n_2), \end{aligned}$$

and the convolution symbol $*$ indicates the 1-D operation on each row, whether input or output. Also the input and output *row sequences* are defined as

$$y_{n_2}(n_1) \triangleq y(n_1, n_2) \quad \text{and} \quad x_{n_2}(n_1) \triangleq x(n_1, n_2).$$

If the coefficient row sequences are 1-D FIR, then we have a conventional 2-D recursive filter, but here we consider the possibility that they may be of infinite support. Note that even 2-D FIR filters are included in the FRF class, by setting $a_0(n_1) = \delta(n_1)$, other $a_{n_2}(n_1) = 0$, and $b_{n_2}(n_1) = b(n_1, n_2) = \text{FIR}$. Upon taking the 2-D Z-transform of (5.2-4), we obtain

$$A_0(z_1)Y(z_1, z_2) = - \sum_{k_2=1}^{L_D} A_{k_2}(z_1)Y(z_1, z_2)z_2^{-k_2} + \sum_{k_2=0}^{L_N} B_{k_2}(z_1)X(z_1, z_2)z_2^{-k_2}.$$

Now we introduce the following rational forms for the 1-D coefficient Z-transforms A_{k_2} and B_{k_2} ,

$$A_{k_2}(z_1) = \frac{N_{k_2}^a(z_1)}{D_{k_2}^a(z_1)} \quad \text{and} \quad B_{k_2}(z_1) = \frac{N_{k_2}^b(z_1)}{D_{k_2}^b(z_1)},$$

where the numerator and denominators are finite-order polynomials in z_1 . They thus constitute rational row operators that can be implemented via recursive filters separately processing the present and most recent L_N input rows and L_D previous output rows. The row coefficients $a_{n_2}(n_1)$ and $b_{n_2}(n_1)$ are then just the impulse responses of these 1-D row operators. The row operator on the current output row A_0 can then

be implemented via inverse filtering, after completion of the sums indicated on the righthand side of (5.2–4).

The overall FRF system function can then be written as

$$H(z_1, z_2) = \frac{\sum_{k_2=0}^{L_N} \frac{N_{k_2}^b(z_1)}{D_{k_2}^b(z_1)} z_2^{-k_2}}{\sum_{k_2=0}^{L_D} \frac{N_{k_2}^a(z_1)}{D_{k_2}^a(z_1)} z_2^{-k_2}}.$$

We pause now for an example.

Example 5.2–3: First-Order FRF with Nonsymmetric Half-Plane Impulse Response Support

Let $L_N = 1$ and $L_D = 1$. Take the feedback row operators A_0 and A_1 with the following system functions

$$A_0(z_1) = \frac{1 + 0.8z_1^{-1}}{1 + 0.9z_1^{-1}} \quad \text{and} \quad A_1(z_1) = 0.8 \frac{2 + z_1^{-1}}{1 + 0.6z_1^{-1}},$$

with ROC containing $\{|z_1| \leq 1\}$ and input row operators B_0 and B_1 with system functions

$$B_0(z_1) = \frac{1 + z_1^{-1}}{1 + 0.7z_1^{-1}} \quad \text{and} \quad B_1(z_1) = 0.7 \frac{1 + z_1^{-1}}{1 + 0.8z_1^{-1}},$$

also with ROC containing $\{|z_1| \leq 1\}$. Then the overall FRF system function is

$$\begin{aligned} H(z_1, z_2) &= \frac{B_0(z_1) + B_1(z_1)z_2^{-1}}{A_0(z_1) + A_1(z_1)z_2^{-1}} \\ &= \frac{\frac{1+z_1^{-1}}{1+0.7z_1^{-1}} + \frac{1+z_1^{-1}}{1+0.8z_1^{-1}}z_2^{-1}}{\frac{1+0.8z_1^{-1}}{1+0.9z_1^{-1}} + \frac{2+z_1^{-1}}{1+0.6z_1^{-1}}z_2^{-1}}. \end{aligned}$$

Observations

1. Each 1-D row operator can be factored into poles inside and poles outside the unit circle and then split into \oplus and \ominus components, where the \oplus component is recursively stable as a right-sided (causal) sequence and the \ominus component is recursively stable as a left-sided (anticausal) sequence.
2. If the inverses $A_0^{-1}(z_1)$ and $B_0^{-1}(z_1)$ are stable and right-sided, then we get non-symmetric half-plane (NSHP) support for the overall FRF impulse response. We get symmetric half-plane (SHP) response support as shown in Figure 5.2–1 when A_0^{-1} and B_0^{-1} are stable and two-sided.
3. It is important to note, in either case, that the impulse responses are not restricted to wedge support as would be the case for a conventional NSHP recursive filter.

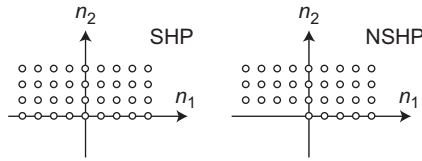
**FIGURE 5.2-1**

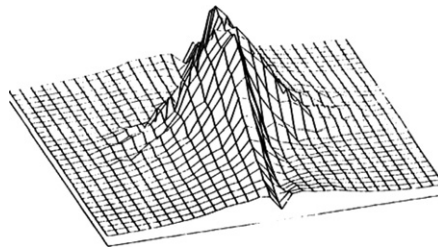
Illustration of FRF support options. Note that FRF impulse response support extends to the complete SHP or NSHP region.

The stability of FRF filters is addressed in [9]. The following filter design example uses a numerical measure of stability as an additional design constraint, thus ensuring that the designed filter satisfies a numerical version of the FRF stability test [9]. The example concerns an FRF design for the ideal 2-D Wiener filter, an optimal linear filter for use in estimating signals in noise, which will be derived and used in Chapter 8 on image processing.

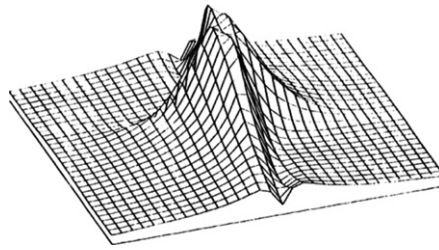
Example 5.2-4: SHP Wiener Filter Design

Using the Levenberg-Marquardt optimization method, the following SHP Wiener filter was designed in [9]. This FRF used 65 coefficients, approximately equally spread over the numerator and denominator and involving five input and output rows. Figures 5.2-2 and 5.2-3 show the ideal and designed magnitude response, and Figures 5.2-4 and 5.2-5 show the respective phase responses.

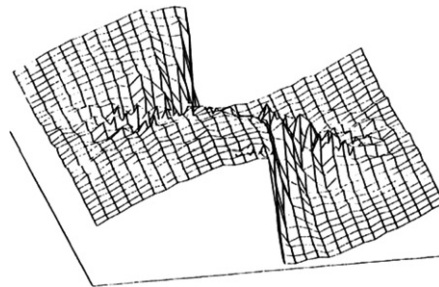
We see that all the main features of the ideal magnitude and phase response of the Wiener filter are achieved by the fully recursive design. This is because the FRF impulse response is not constrained to have support on a wedge of the general causal half-space like the conventional NSHP recursive filter. The measured magnitude MSE was 0.0028 and the phase MSE was 0.042. As seen in [9], the conventional recursive NSHP filter was not able to achieve this high degree of approximation. (See Chapter 8 for more on 2-D Wiener filters and their applications).

**FIGURE 5.2-2**

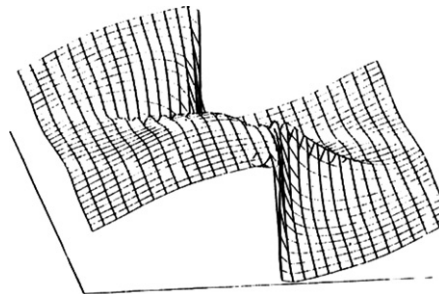
Ideal SHP Wiener filter magnitude (origin in middle).

**FIGURE 5.2-3**

SHP Wiener FRF magnitude (origin in middle).

**FIGURE 5.2-4**

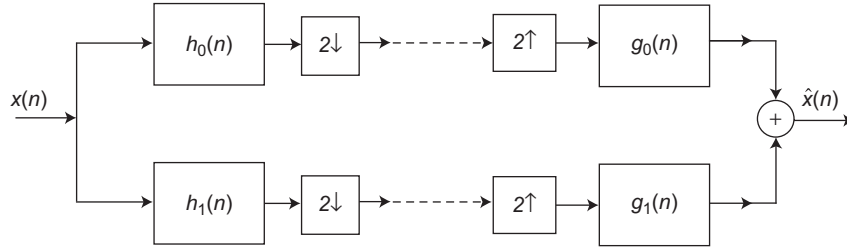
Ideal SHP Wiener phase response (origin in middle).

**FIGURE 5.2-5**

SHP Wiener FRF phase response (origin in middle).

5.3 SUBBAND/WAVELET FILTER DESIGN

Most spatial subband/wavelet or SWT filter design uses the separable product approach. Here, we just discuss some 1-D design methods for the component filters as shown in [Figure 5.3-1](#). Johnston [10] designed pairs of even-length quadrature mirror

**FIGURE 5.3–1**

1-D diagram of analysis/synthesis SWT/ISWT system.

filters (QMF) with linear phase using a computational optimization method. Aliasing was canceled out via the method of Esteban and Galand [11], which makes the synthesis choice $G_0(\omega) \triangleq H_1(\omega - \pi)$ and $G_1(\omega) \triangleq -H_0(\omega - \pi)$, and the lowpass–highpass property of the analysis filterbank was assured via $H_1(\omega) = H_0(\omega - \pi)$. Thus the design can concentrate on the transfer function

$$T(\omega) \triangleq \frac{1}{2} [H_0^2(\omega) - H_0^2(\omega - \pi)].$$

In the perfect reconstruction case, $T(\omega) = 1$, and so $\hat{x}(n) = x(n)$. In the case of Johnston QMFs, perfect reconstruction is not possible, but we can achieve a very good approximation $\hat{x}(n) \simeq x(n)$, as will be seen here. For a given stopband spec ω_s and energy tolerance ϵ , Johnston seeks to minimize

$$\int_0^{\pi/2} [|T(\omega)| - 1]^2 d\omega \text{ subject to constraint } \int_{\omega_s}^{\pi} |H_0(\omega)|^2 d\omega \leq \epsilon,$$

using the Hooke and Jeeves nonlinear optimization algorithm [12]. His design algorithm took advantage of linear phase by working with the real-valued quantity \widetilde{H}_0 defined by the relation

$$\widetilde{H}_0(\omega) \triangleq e^{+j\omega(N-1)/2} H_0(\omega).$$

He used a dense frequency grid to approximate the integrals and used the Lagrangian method to bring in the stopband constraint, resulting in minimization of total error function E :

$$E \triangleq E_r + \lambda E_s,$$

with

$$E_r \triangleq 2 \sum_{\omega_i=0}^{\pi/2} \left[\widetilde{H}_0^2(\omega_i) + \widetilde{H}_0^2(\pi - \omega_i) - 1 \right]^2, \quad (5.3-1)$$

$$E_s \triangleq \sum_{\omega_i=\omega_s}^{\pi} \widetilde{H}_0^2(\omega_i).$$

He thus generated a family of even-length, linear-phase FIR filters parameterized by their length N , stopband ω_s , and stopband tolerance ϵ . Results were reported in terms of transition bandwidth, passband ripple, and stopband attenuation. Johnston originally designed his filters for audio compression. Note that since we trade off E_r versus E_s , we do not get *perfect reconstruction* with these QMF designs; still, the approximation in the total transfer function $T(\omega)$ can be very accurate, i.e., within 0.02 dB, which is quite accurate enough so that errors are not visible in typical image processing displays.

Note that design equation (5.3–1) is only consistent with $T(\omega) = 1/2$, so an overall multiplication by 2 would be necessary when using these Johnston filters.

Example 5.3–1: Johnston's Filter 16C

Here, we look at the step response and frequency response of a 16-tap QMF designed by Johnston [10] using the Hooke and Jeeves nonlinear optimization method. For the 16C filter, the value of $\lambda = 2$ achieved stopband attenuation of 30 dB. The transition bandwidth was 0.10 radians and the achieved overall transmission was flat to within 0.07 dB. Another similar filter, the 16B, achieved 0.02 dB transmission flatness. The step response in Figure 5.3–2 shows about a 5–10% amount of overshoot or ringing.

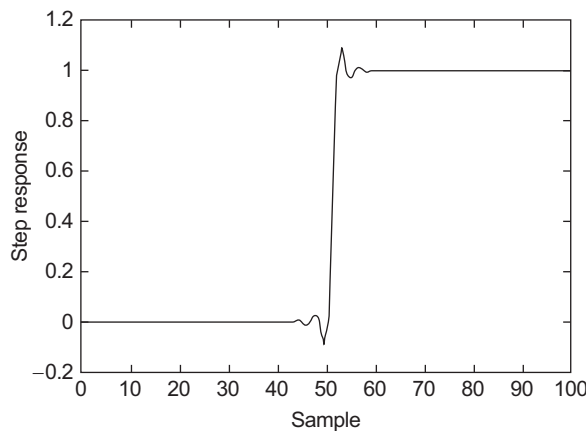
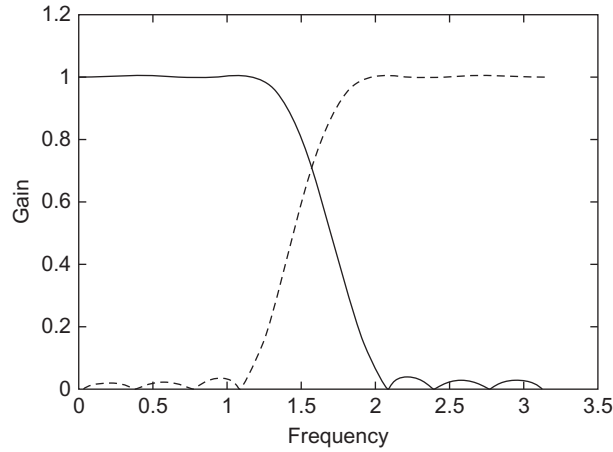


FIGURE 5.3–2

Step response of Johnston 16C linear-phase QMF.

The frequency response in Figure 5.3–3 shows good lowpass and highpass analysis characteristics.

Other filters, due to Simoncelli and Adelson [13], were designed using a similar optimization approach, but using a min–max or l^∞ norm and incorporating a modified $1/|\omega|$ weighting function with bias toward low frequency to better match the human

**FIGURE 5.3-3**

Magnitude frequency response of Johnston 16C filter.

visual system. Their designed filters are useful for both odd and even lengths N due to their use of the alternative QMF condition (4.4–10) and reconstruction equations (4.4–9) in Chapter 4. Notable is their 9-tap filter, which performs quite well in image coding.

Wavelet (Biorthogonal) Filter Design Method

The 1-D subband filter pairs designed by wavelet analysis generally relate to what is called maximally flat design in signal processing literature [14], and the filters are then used for separable 2-D subband analysis and synthesis as discussed previously. The term *biorthogonal* is used in the wavelet literature to denote the case where the analysis filter set $\{h_0, h_1\}$ is different from the synthesis filter set $\{g_0, g_1\}$. Generally the extra freedom in the design of biorthogonal filters results in a more accurate design for the lowpass filters combined with perfect reconstruction. On the other hand, departures from orthogonality generally have a negative effect on coding efficiency. So the best biorthogonal wavelet filters for image coding are usually nearly orthogonal.

Rewriting the perfect reconstruction SWT equations from Section 4.4 of Chapter 4 in terms of Z-transforms, we have the transfer function

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2, \quad (5.3-2)$$

given the aliasing cancellation condition

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0. \quad (5.3-3)$$

In the papers by Antonini et al. [15] and Cohen et al. [16] on biorthogonal subband/wavelet filter design, a perfect reconstruction solution is sought under the

constraint that the analysis and synthesis lowpass filters h_0 and g_0 be symmetric and FIR—that is, linear phase with phase zero. They cancel out aliasing (5.3–3) via the choices [17]

$$H_1(z) = zG_0(-z) \quad \text{and} \quad G_1(z) = z^{-1}H_0(-z). \quad (5.3-4)$$

The transfer function (5.3–2) then becomes

$$H_0(z)G_0(z) + H_0(-z)G_0(-z) = 2,$$

which when evaluated on the unit circle $z = e^{j\omega}$ gives the overall system frequency response

$$H_0(\omega)G_0(\omega) + H_0(\omega - \pi)G_0(\omega - \pi) = 2, \quad (5.3-5)$$

owing to the use of real and symmetric lowpass filters h_0 and g_0 .

It now remains to design h_0 and g_0 to achieve this perfect reconstruction (5.3–2) or (5.3–5). For this they use a spectral factorization method and introduce a prescribed parameterized form for the product $H_0(\omega)G_0(\omega)$ that is known to satisfy (5.3–5) exactly. Since it is desired that both the analysis and reconstruction filters have a high degree of flatness, calling for zeros of the derivative at $\omega = 0$ and π , the following equation for the product H_0G_0 was proposed in [16]:

$$H_0(\omega)G_0(\omega) = \cos^{2K}(\omega/2) \left[\sum_{p=0}^{L-1} \binom{L-1+p}{p} \sin^{2p}(\omega/2) \right], \quad (5.3-6)$$

where $\cos(\omega/2)$ provides zeros at $\omega = \pi$ and the terms $\sin(\omega/2)$ provide zeros at $\omega = 0$. Then actual filter solutions can be obtained with varying degrees of spectral flatness and various lowpass characteristics, all guaranteed to give perfect reconstruction, by factoring the function on the right-hand side of (5.3–6) for various choices of K and L .

The polynomial on the right-hand side of (5.3–6), call it $|M_0(\omega)|^2$, is half of a *power complementary* pair defined by the relation

$$|M_0(\omega)|^2 + |M_0(\omega - \pi)|^2 = 2,$$

which had been considered earlier by Smith and Barnwell [18] for the design of perfect reconstruction orthogonal subband/wavelet filters. A particularly nice treatment of this wavelet design method is presented in Taubman and Marcellin [19].

Example 5.3–2: The 9/7 Cohen-Daubechies-Feauveau (CDF) Filter

In [15], the authors consider several example solutions. The one that works the best in their image coding example corresponds to factors H_0 and G_0 in (5.3–6) that make h_0 and g_0 have nearly equal lengths. A solution corresponding to $K = 4$ and $L = 4$ resulted in the following 9-tap/7-tap (or simply 9/7) filter pair:

n	0	± 1	± 2	± 3	± 4
$2^{-1/2}h_0(n)$	0.602949	0.266864	-0.078223	-0.016864	0.026749
$2^{-1/2}g_0(n)$	0.557543	0.295636	-0.028772	-0.045636	0.0

Figure 5.3–4 shows the lowpass and highpass analysis system frequency response plotted from a 512-point FFT. The corresponding analysis lowpass filter step response is shown in Figure 5.3–5. The synthesis lowpass filter step response is given in Figure 5.3–6. This synthesis step response seems quite similar to that of the Johnston filter in Figure 5.3–2, with perhaps a bit less ringing.

This filter pair has become the most widely used method of SWT analysis and synthesis (ISWT) and is chosen as the default filter for the ISO standard JPEG 2000 [17, 19]. It is regarded for its generally high coding performance and reduced amount of ringing when the upper subband is lost for coding reasons such as quantizing coefficients to 0. This filter set is generally denoted as “CDF 9/7,” for the authors in [16], or often as just “Daubechies 9/7.”

In comparing the QMFs with the wavelet-designed filters, note that the main difference is that the QMFs have a better passband response with a sharper cutoff, which can reduce aliasing error in the lowpass band. The wavelet-designed filters tend to have poorer passband response but less ringing in the synthesis filter step response, and hence less visual artifacts under strong coefficient quantization.

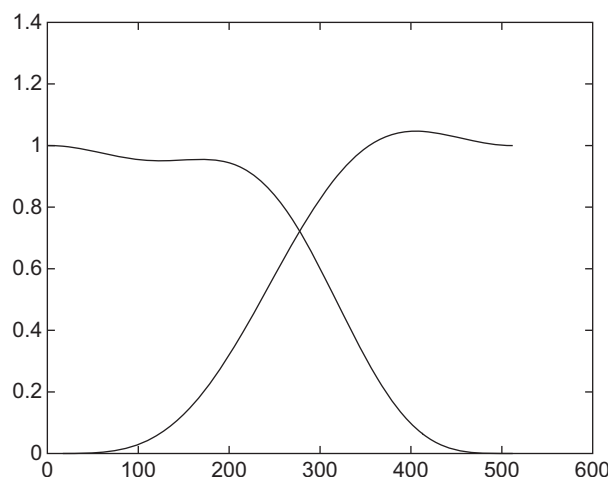
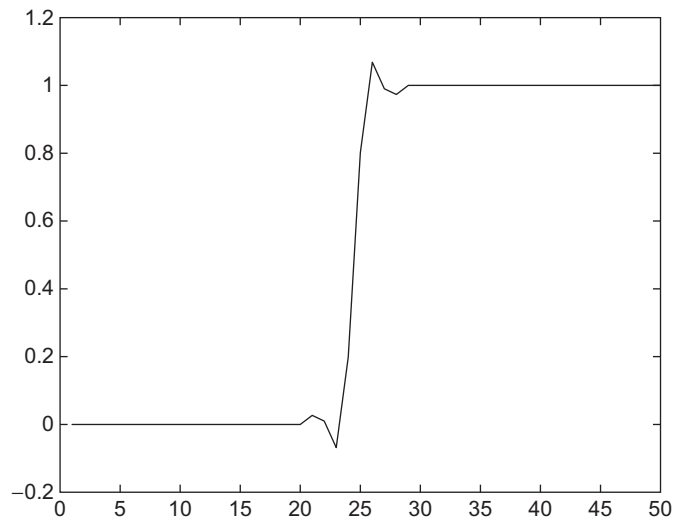
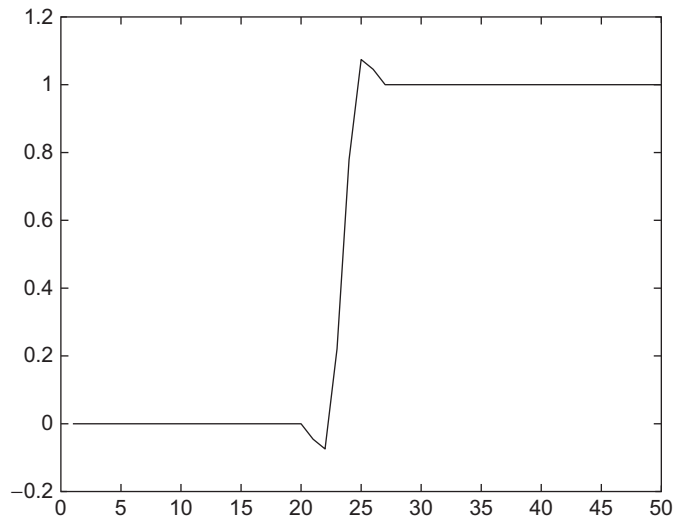


FIGURE 5.3–4

Frequency response of CDF 9/7 analysis filters (512 pt. DFT).

**FIGURE 5.3-5**

Step response of CDF 9/7 analysis lowpass filter.

**FIGURE 5.3-6**

The step response of CDF 9/7 synthesis lowpass filter.

Example 5.3–3: Anti-alias Filtering

Here, we take a 4CIF (704×576) monochrome image and subsample it to CIF (352×288) using two different anti-alias filters. On the left in Figure 5.3–7 is the result using the CDF 9/7 subband/wavelet filter. The image on the right was obtained using the MPEG4-recommended lowpass filter prior to the decimator. The MPEG4 lowpass recommendation has the form

n	0	± 1	± 2	± 3	± 4	± 5	± 6
$64h(n)$	24	19	5	–3	–4	0	2



(a) CDF 9/7 filtered



(b) MPEG4 rec. filtered

FIGURE 5.3–7

An illustration of using two different anti-alias filters for downsampling.

We can see that the image on the right in Figure 5.3–7 is softer, but also that the one on the left has significant spatial frequency aliasing that shows up on the faces of some of the buildings. It should be noted that this image frame from the MPEG test video *City* has an unusual amount of high-frequency data. Most 4CIF images would not alias this badly with the CDF 9/7 lowpass filter.

CONCLUSIONS

In this chapter we introduced 2-D and spatial filter design. We studied FIR filters with emphasis on the window design method but also considered the 1-D to 2-D transformation method and the application of the POCS method to spatial filter design. In the case of IIR or recursive filters, we studied some simple spatial-domain design methods and then briefly overviewed computer-aided design of spatial IIR filters. We introduced FRFs and their design, and saw an example of using an FRF filter

to give a good approximation to the frequency response of an ideal spatial Wiener filter (to be presented in Chapter 8). Finally, we discussed the design problem for subband/wavelet filters as used in a separable SWT.

PROBLEMS

1. A real-valued filter $h(n_1, n_2)$ has support on $[-L, +L] \times [-L, +L]$ and has a Fourier transform $H(\omega_1, \omega_2)$ that is real valued also. What kind of symmetry must h have in the n_1, n_2 plane?

2. Let the real-valued impulse response have quadrantal symmetry:

$$h(n_1, n_2) = h(-n_1, n_2) = h(n_1, -n_2) = h(-n_1, -n_2).$$

What symmetries exist in the Fourier transform? Is it real? How can this be used to ease the computational complexity of filter design?

3. This problem concerns FIR filter design with the constraint of quadrantal symmetry on the impulse response:

$$h(n_1, n_2) = h(-n_1, n_2) = h(n_1, -n_2) = h(-n_1, -n_2).$$

Let the support of this FIR filter h be given as $[-M, +M]^2$.

- (a) Show that the corresponding frequency response has the following symmetry:

$$H(\omega_1, \omega_2) = H(-\omega_1, \omega_2) = H(+\omega_1, -\omega_2) = H(-\omega_1, -\omega_2),$$

and hence the frequency-domain error need only be evaluated in the first quadrant of $[-\pi, +\pi]^2$. (Assume that the ideal frequency response function $H_I(\omega_1, \omega_2)$ has quadrantal symmetry too.)

- (b) Determine a set of design variables a_{k_1, k_2} and basis functions $\phi_{k_1, k_2}(\omega_1, \omega_2)$ to efficiently express the frequency response

$$H(\omega_1, \omega_2) = \sum_{(k_1, k_2) \in \mathcal{R}} a_{k_1, k_2} \phi_{k_1, k_2}(\omega_1, \omega_2),$$

where $\mathcal{R} = [0, M]^2$. The a_{k_1, k_2} should be expressed in terms of the $h(k_1, k_2)$, and the ϕ_{k_1, k_2} should be expressed in terms of the $\cos \omega_1 k_1$ and $\cos \omega_2 k_2$.

4. A certain *ideal* lowpass filter with cutoff frequency $\omega_c = \frac{\pi}{2}$ has impulse response

$$h_d(n_1, n_2) = \frac{1}{\sqrt{n_1^2 + n_2^2}} J_1 \left(\frac{\pi}{2} \sqrt{n_1^2 + n_2^2} \right).$$

- (a) What is the passband gain of this filter? (Hint: $\lim_{x \rightarrow 0} \frac{J_1(x)}{x} = \frac{1}{2}$)

- (b) We want to design an $N \times N$ -point, linear phase, FIR filter $h(n_1, n_2)$ with *first quadrant* support using the ideal function $h_d(n_1, n_2)$ given above. We choose a *separable* 2-D Kaiser window with parameter $\beta = 2$. The continuous-time 1-D Kaiser window is given as

$$w_\alpha(t) = \begin{cases} I_0(\beta\sqrt{1 - (t/\tau)^2})/I_0(\beta), & |t| < \tau, \\ 0, & \text{else.} \end{cases}$$

Write an expression for the filter coefficients of h —i.e., $h(n_1, n_2) : n_1 = 0, \dots, N-1, n_2 := 0, \dots, N-1$.

- (c) If we use a 512×512 -point row–column FFT to approximately implement the preceding designed filter h for a 512×512 -pixel image,
- First, evaluate the number of complex multiplies used as a function of M for an $M \times M$ image, assuming the component 1-D FFT uses the Cooley-Tukey approach. (The 1-D Cooley-Tukey FFT algorithm needs $\frac{1}{2}M \log_2 M$ complex multiplies when M is a power of 2 approach.) Specialize your result to $M = 512$.
 - What portion of the output will be the correct (*linear convolution*) result? Specify which pixels will be correct.

5. We want to design an $N \times N$ -point, *linear phase*, FIR filter $h(n_1, n_2)$ with *first quadrant support* using the ideal function $h_i(n_1, n_2)$ given as

$$h_i(n_1, n_2) = \frac{1}{4} \frac{J_1\left(\frac{\pi}{2}\sqrt{n_1^2 + n_2^2}\right)}{\sqrt{n_1^2 + n_2^2}}.$$

We choose a *separable* 2-D Kaiser window with parameter $\beta = 2$. The continuous time 1-D Kaiser window is given as

$$w_\alpha(t) = \begin{cases} I_0(\alpha\sqrt{1 - (t/\tau)^2})/I_0(\alpha), & |t| < \tau, \\ 0, & \text{else.} \end{cases}$$

- (a) Write down the equation for the filter coefficients of h —i.e., $h(n_1, n_2) : n_1 = 0, \dots, N-1, n_2 = 0, \dots, N-1$. Assume N is odd.
- (b) Use MATLAB to find the magnitude of the frequency response of the designed filter. Plot the result for $N = 11$.
6. Let $0 < \omega_p < \pi$, and define the 1-D ideal filter response

$$H_I(\omega) \triangleq \begin{cases} 1, & |\omega| \leq \frac{1}{2}(\omega_s + \omega_p), \\ 0, & \text{else,} \end{cases} \quad \text{on } [-\pi, +\pi].$$

Then define $H_{I1}(\omega) \triangleq W_1(\omega) \otimes H_I(\omega)$, where \otimes denotes periodic convolution and where

$$W_1(\omega) \triangleq \begin{cases} \frac{1}{\omega_s - \omega_p}, & |\omega| \leq \frac{1}{2}(\omega_s - \omega_p), \\ 0, & \text{else,} \end{cases} \quad \text{on } [-\pi, +\pi], \text{ with } \pi > \omega_s > \omega_p.$$

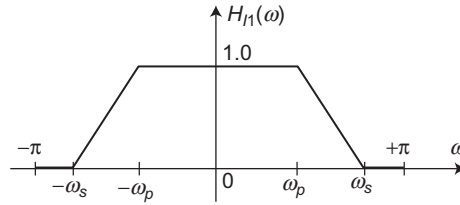


FIGURE 5.P-1

Alternative ideal filter response.

$H_{I1}(\omega)$ is as shown in Figure 5.P-1.

- (a) Find the impulse response $h_{I1}(n)$. Note that it is of infinite support.
 - (b) Define and sketch the 2-D ideal filter response $H_{I1}(\omega_1, \omega_2) \triangleq H_{I1}(\omega_1)H_{I1}(\omega_2)$ and give an expression for its impulse response $h_{I1}(n_1, n_2)$.
 - (c) Find the so-called l^2 or minimum least-squares error approximation to the ideal spatial filter you found in part (b), with support $[-N, +N]^2$.
7. Modify the MATLAB program `MacDesFilt.m` (available at the book's Web site) to design a nearly circular highpass filter with stopband edge $\omega_s = 0.1$ and $\omega_p = 0.3$, and apply the filter to the *Lena* image. (Note: This requires MATLAB with image processing toolbox.) Is the output what you expected? Try scaling and shifting the output y to make it fit in the output display window $[0, 255]$. (Hint: The output scaling $y = 128 + 2 * y$ works well.)
 8. In deriving the McClellan transformation method, we need to find the $a'(n)$ coefficients in (5.1-2) in terms of the $a(n)$ coefficients in (5.1-1). In this problem, we work out the case for $M = 4$.
 - (a) Use $T_n(x)$ for $n = 0, 4$ to evaluate $\cos n\omega$ in terms of powers of $\cos^k \omega$ for $k = 0, 4$.
 - (b) Substitute these expressions into (5.1-1) and determine $a'(n)$ in terms of $a(n)$ for $n = 0, 4$.
 - (c) Solve the equations in part (b) for $a(n)$ in terms of $a'(n)$.
 9. In the design by transformation method due to McClellan, assume the 1-D linear-phase (type I or N odd) FIR filter has coefficients $h(0), h(1), \dots, h(N-1)$.
 - (a) Express the resulting 2-D filter's coefficients $h(n_1, n_2)$ in terms of the $a'(n)$ derived from the 1-D filter coefficients $h(n)$. Do this for the general transformation $F(\omega_1, \omega_2)$ in terms of parameters A, B, C, D , and E as given in (5.1-3). In your expression for $h(n_1, n_2)$, you may use the *convolution power* defined as

$$x^{*k} \triangleq x * x^{*(k-1)}, \quad k > 1 \quad \text{with} \quad x^{*1} \triangleq x.$$

- (b) Devise an implementation method that minimizes the number of nontrivial multiplications assuming that $A-E$ are negative powers of 2.

10. Let the filter h with first quadrant impulse response support have system function

$$H(z_1, z_2) = \frac{b_{00}}{1 + a_{10}z_1^{-1} + a_{01}z_2^{-1} + a_{11}z_1^{-1}z_2^{-1}}.$$

- (a) Find the coefficients $\{b_{00}; a_{10}, a_{01}, a_{11}\}$ such that the impulse response h agrees with the four prescribed values:

$$\begin{aligned} h(0,0) &= 1.0, & h(1,0) &= 0.8, \\ h(0,1) &= 0.7, & h(1,1) &= 0.6. \end{aligned}$$

- (b) Determine whether the resulting filter is stable or not. (Hint: Try the root mapping method.)

11. In 2-D filter design, we often resort to a general optimization program such as Fletcher-Powell to perform the design. Such programs often ask for analytic forms for the partial derivatives of the specified error function with respect to each of the filter coefficients.

Consider a simple recursive filter with frequency response

$$H(\omega_1, \omega_2) = \frac{a}{1 - be^{-j\omega_1} - ce^{-j\omega_2}},$$

where the three filter parameters a, b , and c are real variables.

- (a) Find the three partial derivatives of H with respect to each of a, b , and c , for fixed ω_1, ω_2 .
 (b) Find the corresponding partial derivatives of the conjugate function H^* .
 (c) Choose a weighted mean-square error function as

$$\mathcal{E} = \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} W(\omega) |I(\omega) - H(\omega)|^2 d\omega,$$

for given ideal frequency-response function I and positive weighting function W . Find the partial derivatives $\partial\mathcal{E}/\partial a$, $\partial\mathcal{E}/\partial b$, and $\partial\mathcal{E}/\partial c$, making use of your results in parts (a) and (b). (Hint: Rewrite $|A|^2 = AA^*$ first.) Express your answers in terms of W, I , and H .

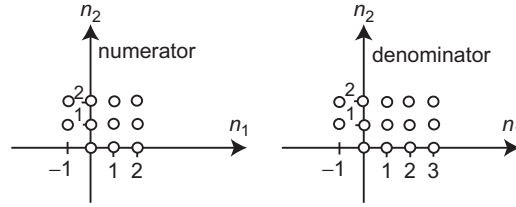
12. Consider a zero-phase or two-pass filter design with NSHP recursive filters

$$H(\omega_1, \omega_2) \triangleq H_{\oplus+}(\omega_1, \omega_2)H_{\ominus-}(\omega_1, \omega_2) = |H(\omega_1, \omega_2)|^2,$$

where $H_{\ominus-}(\omega_1, \omega_2) \triangleq H_{\oplus+}^*(\omega_1, \omega_2)$. Equivalently, in the spatial domain

$$h(n_1, n_2) = h_{\oplus+}(n_1, n_2) * h_{\ominus-}(n_1, n_2).$$

Let the NSHP filter $H_{\oplus+}(\omega_1, \omega_2)$ have real coefficients with numerator support and denominator support as indicated in Figure 5.P-2.

**FIGURE 5.P-2**

Numerator and denominator coefficient support region indicated by open circles.

- (a) Find the spatial support of $h_{\oplus+}(n_1, n_2)$ assuming that this filter is stable in the $+n_1, +n_2$ direction.
 - (b) State the spatial support of $h_{\ominus-}(n_1, n_2)$.
 - (c) What is the support of $h(n_1, n_2)$?
13. Show that the biorthogonal subband/wavelet constraint equation (5.3-4) achieves alias cancellation in (5.3-3) and that the transfer function (5.3-2) becomes

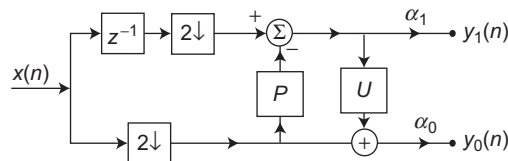
$$H_0(z)G_0(z) + H_0(-z)G_0(-z) = 2,$$

which when evaluated on the unit circle $z = e^{j\omega}$ becomes

$$H_0(\omega)G_0(\omega) + H_0(\omega - \pi)G_0(\omega - \pi) = 2,$$

owing to the use of real and symmetric lowpass filters h_0 and g_0 .

14. The *lifted* SWT is defined starting from the *lazy* SWT, which just separates the input sequence x into even and odd indexed terms. In the lifted SWT, this is followed by *prediction* and *update* steps as specified by the operators P and U , respectively, as shown in Figure 5.P-3. The output multipliers α_0 and α_1 may be needed for proper scaling.

**FIGURE 5.P-3**

An illustration of the lifted SWT.

- (a) Find the corresponding prediction and update operators for the SWT that uses the Haar filter set

$$h_0(n) = \delta(n) + \delta(n-1),$$

$$h_1(n) = \delta(n) - \delta(n-1).$$

- (b) A key property of *lifting* is that the operators P and U are not constrained at all. Show the ISWT for the lifted transform in Figure 5.P-3 by starting from the right-hand side of the figure and first undoing the update step and then undoing the prediction step. Note that this is possible even for nonlinear operators P and U .
- (c) Can you do the same for the LeGall-Tabatabai (LGT) 5/3 analysis filter set

$$h_0(n) = \left\{ -\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8} \right\}$$

$$h_1(n) = \left\{ -\frac{1}{2}, 1, -\frac{1}{2} \right\}?$$

The corresponding synthesis 5/3 filter set is

$$g_0(n) = \left\{ \frac{1}{2}, 1, \frac{1}{2} \right\}$$

$$g_1(n) = \left\{ -\frac{1}{8}, -\frac{1}{4}, \frac{3}{4}, -\frac{1}{4}, -\frac{1}{8} \right\}.$$

Here, h_0 and g_0 are centered on $n = 0$, while h_1 is centered at $n = -1$ and g_1 is centered on $n = +1$. Reference to [17] and/or [19] may be helpful here.

REFERENCES

- [1] J. F. Kaiser, "Nonrecursive Digital Filter Design Using I_0 -sinh Window Function," *Proc. IEEE Sympos. on Circuits and Systems*, pp. 20-23, 1974.
- [2] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [3] A. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd Ed. Prentice-Hall, Englewood Cliffs, NJ, 1999.
- [4] J. H. McClellan and D. S. K. Chan, "A 2-D FIR Filter Structure Derived from the Chebyshev Recursion," *IEEE Trans. Circuits Systems*, vol. CAS-24, pp. 372-378, July 1977.
- [5] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [6] H. Stark and Y. Yang, *Vector Space Projections*, John Wiley, New York, 1998.
- [7] A. Abo-Taleb and M. M. Fahmy, "Design of FIR Two-Dimensional Digital Filters by Successive Projections," *IEEE Trans. Circuits and Sys.*, vol. CAS-31, pp. 801-805, September 1984.

- [8] D. B. Harris and R. M. Mersereau, "A Comparison of Algorithms for Minimax Design of Two-Dimensional Linear Phase FIR Digital Filters," *IEEE Trans. Accoust., Speech, and Signal Process.* vol. ASSP-25, pp. 492–500, December 1977.
- [9] J.-H. Lee and J. W. Woods, "Design and Implementation of Two-Dimensional Fully Recursive Digital Filters," *IEEE Trans. Accoust., Speech, and Signal Process.* vol. ASSP-34, pp. 178–191, February 1986.
- [10] J. D. Johnston, "A Filter Family Designed for Use in Quadrature Mirror Filter Banks," *Proc. IEEE Intl. Conf. Accoust., Speech, and Signal Process (ICASSP)*, pp. 291–294, Denver, CO, 1980.
- [11] D. Esteban and C. Galand, "Application of Quadrature Mirror Systems to Split Band Voice Coding Schemes," *Proc. ICASSP*, pp. 191–195, May 1977.
- [12] Hooke & Jeeves, "Direct Search Solution of Numerical and Statistical Problems," *Journal of the ACM*, vol. 8, pp. 212–229, April 1961.
- [13] E. P. Simoncelli and E. H. Adelson, "Subband Transforms," Chapter 4 in *Subband Image Coding*, Ed. J. W. Woods, Kluwer, Dordrecht, NL, pp. 143–192, 1991.
- [14] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [15] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform," *IEEE Trans. Image Process.*, vol. 1, pp. 205–220, April 1992.
- [16] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal Bases of Compactly Supported Wavelets," *Comm. Pure and Applied Math.*, vol. XLV, pp. 485–560, 1992.
- [17] M. Rabbani and R. Joshi, "An Overview of the JPEG 2000 Still Image Compression Standard," *Signal Process Image Comm.*, vol. 17, pp. 3–48, 2002.
- [18] M. J. T. Smith and T. P. Barnwell III, "Exact Reconstruction for Tree-structured Subband Coders," *IEEE Trans. Accoust., Speech, and Signal Process.* vol. ASSP-34, pp. 431–441, June 1986.
- [19] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression, Fundamentals, Standards, and Practice*, Kluwer Academic Publishers, Norwell, MA, 2002.

Image Perception and Sensing

6

This chapter covers some basics of image processing that are necessary in the applications in later chapters. We will discuss light, sensors, and the human visual system. We will talk about spatial and temporal properties of human vision and present a basic spatiotemporal frequency response of the eye–brain system. This information, interesting in its own right, will be useful for the design and analysis of image and video signal processing systems, which produce images and video to be seen by human observers.

6.1 LIGHT AND LUMINANCE

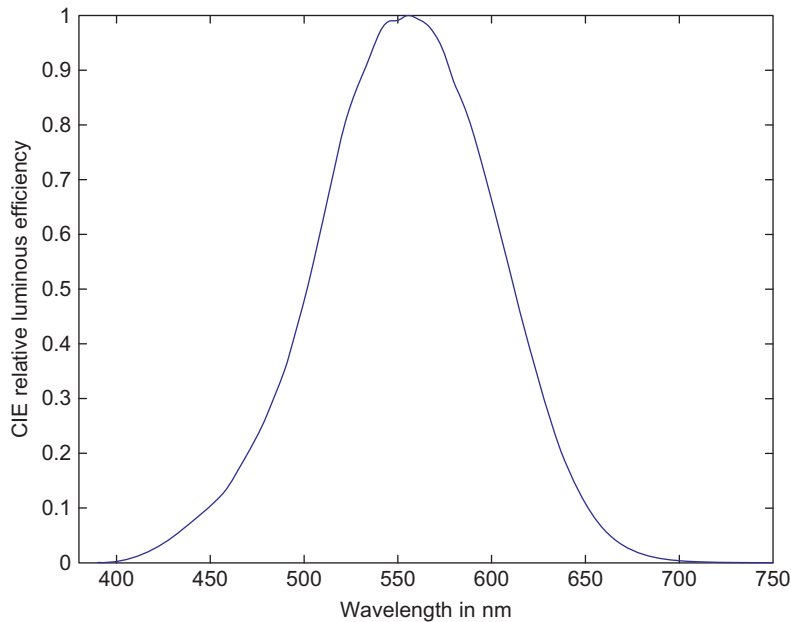
We use the notation $p_i(x, y, t, \lambda)$ for incident radiant flux or light intensity (incident power) as a function of position (x, y) and time t at the spatial wavelength λ . The *human visual system* (HVS) does not perceive this radiant flux directly. A *relative luminous efficiency* $v(\lambda)$ has been defined to relate our perceived *luminance* (brightness) to the incoming radiant flux

$$l(x, y, t) \triangleq K \int_0^{\infty} p_i(x, y, t, \lambda) v(\lambda) d\lambda,$$

where the constant $K = 685$ lumens/watt and the luminance l is expressed in lumens/square-meter [1]. Suppressing the dependence on space and time, we can write this equation more simply as

$$l \triangleq K \int_0^{\infty} p_i(\lambda) v(\lambda) d\lambda$$

and think of it as defining equivalence classes of radiant intensity functions $p_i(\lambda)$ that all have the same luminance—i.e., all look equally bright to a human observer. We can think of the relative luminous efficiency $v(\lambda)$ as a kind of a *wavelength filter* response of the human eye. As such it would be expected to vary somewhat

**FIGURE 6.1-1**

CIE 1929 relative luminous efficiency (standard observer) curve $v(\lambda)$.

from person to person, but a so-called *standard observer* was defined in 1929 by the Commission Internationale de L'eclairage (International Commission on Illumination), known as the CIE, and is plotted in Figure 6.1-1, thereby defining a *standard luminance* with symbol Y .

The question naturally arises as to how such a relative luminous efficiency function $v(\lambda)$ can be measured. Actually, it was done using a clever brightness matching experiment where volunteer viewers were shown reference and test images on a split screen in a narrow viewing angle of 2° and asked to report when they just noticed a brightness difference between the two nearby spectral pairs [2]. Since the original experiments, various refinements in the method have yielded modest improvements in accuracy, principally in the blue or short wavelength area. However, the original CIE curve is still used as the reference for most *photometric* work, *photometry* being the science of measuring the human visual effect of various light stimuli.

An alternative to this split-screen approach is the flicker method, wherein two different spectra are presented to the viewer alternately and at an intermediate speed, and flickering can be seen if the colors are different. The intensities are varied until the viewer does not perceive the flickering anymore. Another question is how much variation there is in this standard response between individuals. The answer

is that there is a modest variation, which has been quantified in the experimental data plots from which the standard observer curve $v(\lambda)$ was created. However, it has been found that the standard curve can predict reasonably well how the HVS will respond.

Note that luminance is a quantity averaged over wavelength¹ and, as such, is called a monochrome or gray-level measure of light intensity. Looking at Figure 6.1–1, normalized to 1, we see that the response of human vision peaks around 555 nanometers (nm), which is *green* light, and has a range of about 420 to 670 nm, with 450 nm called *blue* and 650 nm called *red*. The low end of visibility is *violet*, and the range below the visible is called *ultraviolet*. At the high end we have *red* light, and beyond this end of the visible region is called *infrared*.

We can also define similar response functions for various sensors; suppressing the spatiotemporal dependency on x, y, t , we have

$$I \triangleq K \int_0^{\infty} p_i(\lambda) s_{BW}(\lambda) d\lambda,$$

where s_{BW} is the black–white (gray-level) sensitivity of the imaging device. Examples of image sensors are charge-coupled devices (CCD), various types of image pickup tubes, and coupled metal-oxide semiconductor (CMOS) image sensors. By inserting *optical filters* (wavelength filters) in front of these devices, we can get different responses from various color regions or channels. It is common in today’s cameras to employ optical filters to create three color channels—red, green, and blue—to match what is known about the color perception of the HVS (i.e., the human eye–brain system). Analogous spectral response functions are used to characterize image displays. We will return to color later in this chapter.

6.2 STILL IMAGE VISUAL PROPERTIES

Here, we look at several visual phenomena that are important in image and video perception. First, Weber’s law defines contrast and introduces the concept of *just-noticeable difference* (JND). We then present the contrast sensitivity function, also called the visual magnitude transfer function (MTF), as effectively the human visual frequency response. We then introduce the concept of spatial adaptation of contrast sensitivity.

¹ *Wavelength and spatial frequency*: Image and video sensors record energy (power \times time) on a scale of micrometers or more (for typical 1- to 5-cm sensor chip) and so are not sensitive to the oscillations of the electromagnetic field at the nanometer level. Our spatial frequency is a variation of this average power on the larger scale. So there is no interaction between the wavelength of the light and the spatial frequency of the image, at least for present-day sensors.

Weber's Law

Psychovisual researchers early on found that the eye–brain response to a uniform step in intensity is not the same over the full range of human perception. In fact, they found that the just-noticeable percent is nearly constant over a wide range. This is known now as Weber's law. Writing I for the incident intensity (or luminance) and ΔI for the just-noticeable change, we have

$$\frac{\Delta I}{I} \approx \text{constant},$$

with the constant value in the range [.01, .03], and this value holds constant for at least three decades in $\log I$, as sketched in Figure 6.2–1. We note that Weber's law says we are more sensitive to light intensity changes in low light levels than in strong ones.

A new quantity called *contrast* was then defined via

$$\Delta C = \frac{\Delta I}{I},$$

which in the limit of small Δ becomes the differential equation $dC/dI = I^{-1}$, which integrates to

$$C = \ln(I/I_0). \quad (6.2-1)$$

Sometimes we refer to signals I as being in the *intensity domain*, and the nonlinearly transformed signals C as being in the *contrast* or *density domain*, the latter name coming from a connection with photographic film, whose optical density possesses a similar nonlinearity in response to light intensity (see **Film** in Section 6.6). We note that the contrast domain is most likely the better choice for quantizing a value with uniform step size. This view should be tempered with an understanding of how the psychovisual data were taken to verify Weber's law.

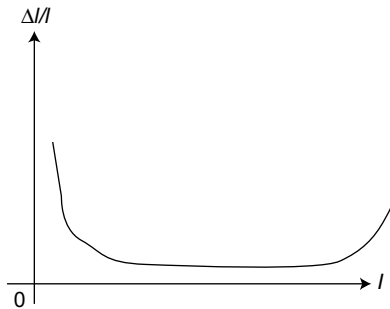


FIGURE 6.2–1

Average just-resolvable intensity difference versus background intensity.

With reference to Figure 6.2–2, human subjects were presented with a disk of incremented intensity $I + \Delta I$ in a uniform background of intensity I and were asked if they could notice the presence of the disk or not. These results were then averaged over many observers, whereby the threshold effect seen in Weber's law was found. The actual JND threshold is typically set at the 50% point in these distributions. Figure 6.2–3 illustrates the concept of such a JND test, showing five small disks of increasing contrast (+2,+4,+6,+8,+10) on three different backgrounds, with contrasts 50, 100, and 200 on the 8-bit range [0,255].

Contrast Sensitivity Function

Another set of psychovisual experiments has determined what has come to be regarded as the spatial frequency response of the HVS. In these experiments, a uniform plane wave is presented to viewers at a given distance, and the angular period

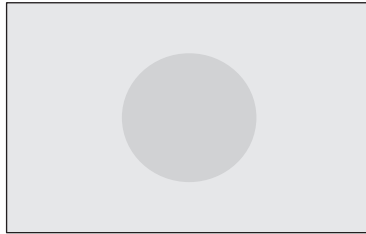


FIGURE 6.2–2

A stimulus that can be used to test Weber's law.



FIGURE 6.2–3

Equal increments at three different brightness values—50, 100, 200—on range [0,255].

of the image focused on their retina is calculated. The question is at what intensity does the plane wave (sometimes called an optical grating) first become visible. The researcher then plots this value as a function of angular spatial frequency, expressed in units of cycles/degree. These values are then averaged over many observers to come up with a set of threshold values for the so-called *standard observer*. The reciprocal of this function is then taken as the human visual frequency response and called the *contrast sensitivity function* (CSF), as seen plotted in Figure 6.2–4, reprinted from [3, 4]. Of course, there must be a uniform background used that is of a prescribed value, on account of Weber’s law. Otherwise the threshold would change. Also, for very low spatial frequencies, the threshold value should be given by Weber’s observation, since he used a plain or *flat* background. The CSF is based on the assumption that the above-threshold sensitivity of the HVS is the same as the threshold sensitivity. In fact, this may not be the case, but it is the current working assumption. Another objection could be that the threshold may not be the same with two stimuli present (i.e., the sum of two different plane waves). Nevertheless, with the so-called *linear hypothesis*, we can weight a given disturbance presented to the HVS with a function that is the reciprocal of these threshold values and effectively normalize them. If the overall intensity is then gradually reduced, all the gratings will become invisible (not noticeable to about 50% of human observers with normal acuity) at the same point. In that sense then, the CSF is the frequency response of the HVS.

An often-referenced formula that well approximates this curve was obtained by Mannos and Sakrison [5] and is expressed in terms of the frequency $f_r \triangleq \sqrt{f_1^2 + f_2^2}$ in cycles/degree,

$$H(f_r) = A \left(\alpha + \frac{f_r}{f_0} \right) \exp - \left(\frac{f_r}{f_0} \right)^\beta, \quad f_r \geq 0,$$

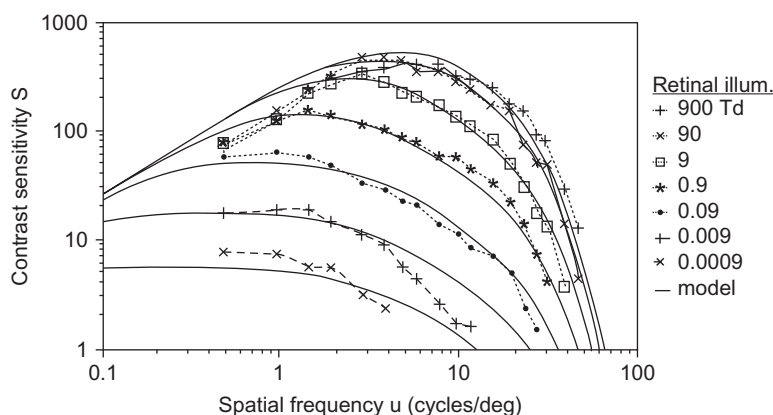


FIGURE 6.2–4

Contrast sensitivity measurements of van Ness and Bouman. (JOSA, 1967 [4])

with $A = 2.6$, $\alpha = 0.0192$, $f_0 = 8.772$, and $\beta = 1.1$. (Here, $f = \omega/2\pi$.) In this formula, the peak occurs at $f_r = 8$ (cycles/degree) and the function is normalized to maximum value 1. A linear amplitude plot of this function is given in Figure 6.2–5. Note that sensitivity is quite low at zero spatial frequency, so some kind of spatial structure or texture is almost essential if a feature is to be noticed by our HVS.

Local Contrast Adaptation

In Figure 6.2–6 we see a pair of small squares, one on a light background and the other on a dark background. While the small squares appear to have different gray levels, in fact the gray levels are the same. This effect occurs because the HVS adapts to surrounding brightness levels when it interprets the brightness of an object.

There is also a local contrast adaptation wherein the JND moves upward as the background brightness moves away from the average contrast of the object. Such a test can be performed via the approach sketched in Figure 6.2–7, where we note that the small central square is split and slightly darker on the left. For most people, this effect is more evident from the display on the left, where the contrast with the local background is only slight. However, on the right the large local contrast with the local

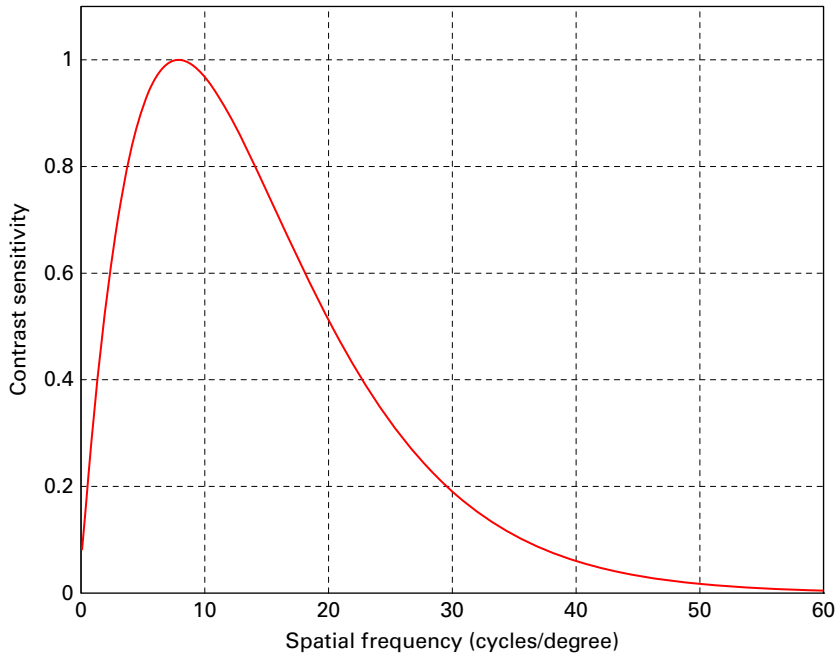
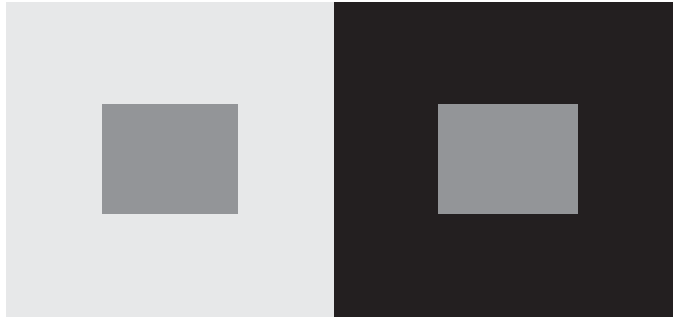


FIGURE 6.2–5

Plot of the Mannos and Sakrison CSF [5].

**FIGURE 6.2-6**

An illustration of the local adaptation property of the HVS.

**FIGURE 6.2-7**

Illustration of dependence of JND on local background brightness.

background makes this effect harder to perceive. Effectively, this means that the JND varies somewhat with local contrast, or said another way, there is some kind of local masking effect.

6.3 TIME-VARIANT HUMAN VISUAL SYSTEM PROPERTIES

The same kind of psychovisual experiments used in the preceding discussion can be employed to determine the visibility of time-variant features. By employing a spatial plane wave grating and then modulating it sinusoidally over time, the contrast sensitivity of the HVS can be measured in a spatiotemporal sense. Plots of experiments by Kelly (1979) [6] and (1971) [7], as reprinted by Barten [3], are shown here. The first, [Figure 6.3-1](#), shows slices of the spatiotemporal CSF plotted

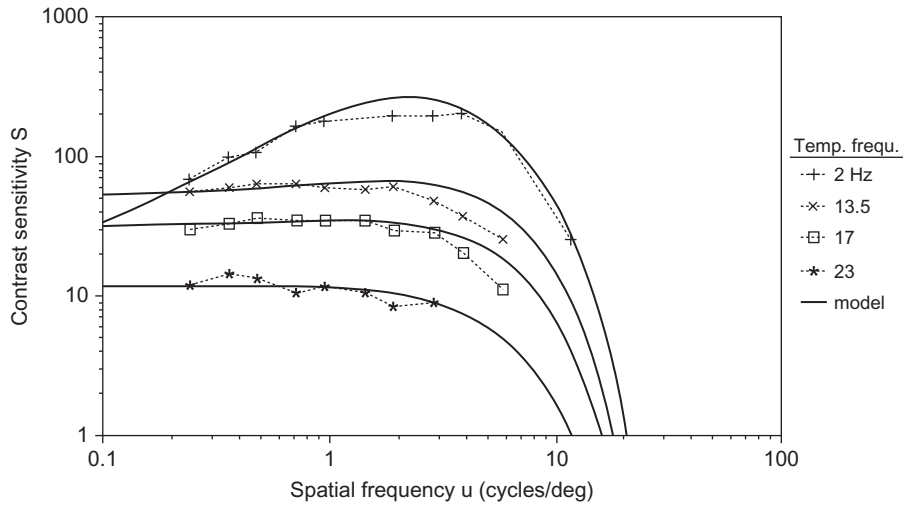


FIGURE 6.3-1

Spatiotemporal CSF from Kelly. (JOSA, 1979 [6])

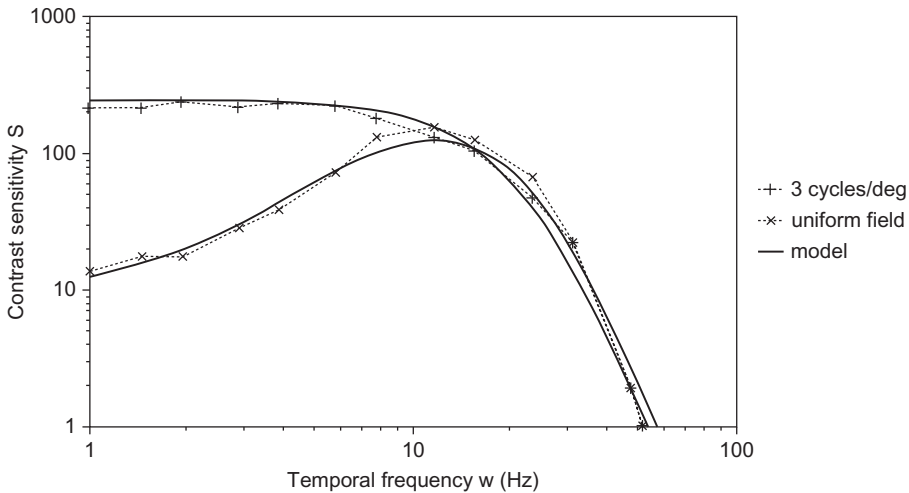


FIGURE 6.3-2

Temporal CSF with spatial parameter from Kelly. (JOSA, 1971 [7])

versus spatial frequency, with the slices at various temporal frequencies, over the range 2 to 23 Hz. We see that maximum response occurs around 8 cycles/degree (cpd) at 2 Hz. Also, we see no bandpass characteristic at the higher temporal frequencies. Figure 6.3-2 shows slices of the CSF as measured by Kelly (1971) where

there are two slices at spatial frequencies of 0 and 3 cpd, plotted versus temporal frequency. Again, we see the bandpass characteristic at the very low spatial frequency, while there is a lowpass characteristic at the higher spatial frequencies. Note that the highest contrast sensitivity is at temporal frequency 0 and spatial frequency 3 cpd.

A 3-D perspective log plot of the spatiotemporal CSF was shown in Lambrecht and Kunt [8] and is reprinted in Figure 6.3–3. Note that these 3-D CSFs are not separable functions. Note also that there is an order of magnitude difference between the sensitivity at DC or origin and that at the peak of maybe about 10 Hz and 6 cpd.

A summary comment is that the HVS seems to be sensitive to change, either spatially or temporally, and that is what tends to get noticed. Lesser variations tend to be ignored. These results can be used to perceptually weight various error criteria for image and video signal processing, to produce a result more visually pleasing. An excellent source for further study is the thesis monograph of Barten [3]. Human visual perception is also covered in the review Chapter 8.2 by Pappas et al. in *Handbook of Image and Video Processing*, 2nd Edition [9]. Basic image processing techniques are also covered by Bovik in Chapter 2.1 of the same handbook.

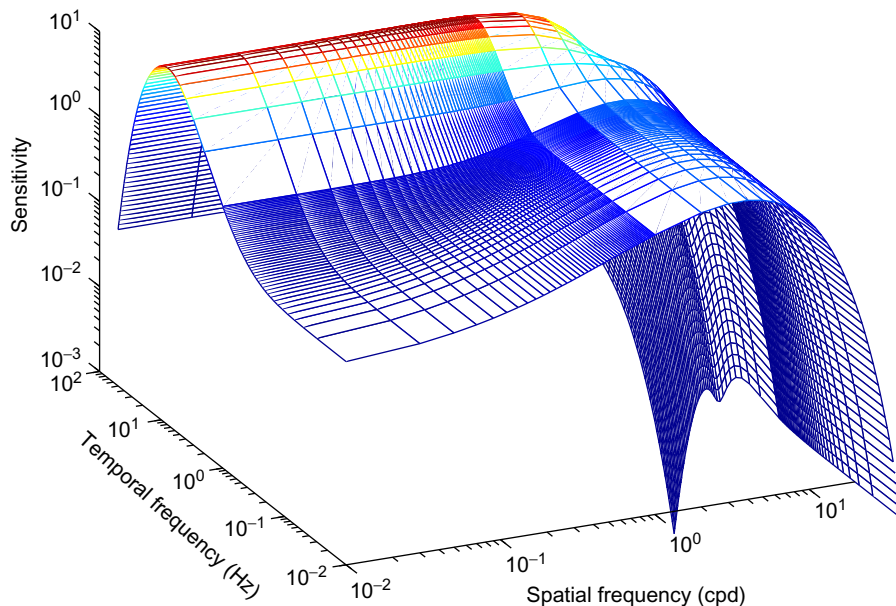


FIGURE 6.3–3

A perspective plot of a spatiotemporal CSF from Lambrecht and Kunt (*Image Communication* 1998 [8]) (normalized to 10).

6.4 COLOR

The human eye itself is composed of an *iris*, a *lens*, and an imaging surface or *retina*. The purpose of the iris is to let in the right amount of light to avoid either saturation or under-illumination. The lens focuses this light on the retina, which is composed of individual *cones* and *rods*. The rods are spread over a very large area of the retina but at a considerably lower spatial density than the cones. These rods are of one type and therefore can only provide a monochrome response. They are very sensitive to light, though, and provide our main perception at dim light levels, such as at night or in a very darkened room. This is why we do not perceive color outdoors at night.

The cones are deployed at higher density in a small central part of the retina called the *fovea*. These cones are of three color efficiencies, responding roughly to red, green, and blue. Labeling these luminous efficiencies as $s_R(\lambda)$, $s_G(\lambda)$, and $s_B(\lambda)$, we see the result of psychovisual experiments, sketched in Figure 6.4–1, with the green and blue response being fairly distinct, and the red response perhaps being more appropriately called yellow-green, overlapping considerably with the green.

We can then characterize the average human cone response to an incident flux distribution $p_i(\lambda)$ as

$$R = \mathcal{K} \int_0^{\infty} p_i(\lambda) s_R(\lambda) d\lambda, \quad (6.4-1)$$

$$G = \mathcal{K} \int_0^{\infty} p_i(\lambda) s_G(\lambda) d\lambda, \text{ and} \quad (6.4-2)$$

$$B = \mathcal{K} \int_0^{\infty} p_i(\lambda) s_B(\lambda) d\lambda. \quad (6.4-3)$$

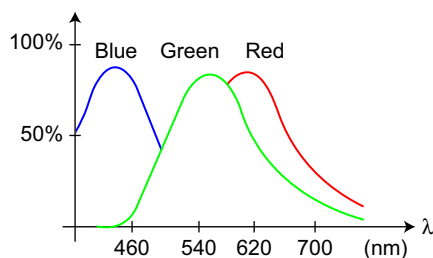


FIGURE 6.4–1

Sketch of average sensitivity functions of three types of color receptors (cones) in the fovea of the human eye.

These three equations are a linear model for color that is in accord with experimentally observed Grassmann “laws” [10, 11] of color perception. As such, we have a 3-D subset of a function space with three basis functions, $s_R(\lambda)$, $s_G(\lambda)$, and $s_B(\lambda)$. All incident power densities $p_i(\lambda)$ that have the same R , G , and B values as calculated in (6.4-1 and 6.4-3) will have the same “color.” This set of power densities is then said to be *metameric*, or of the same color [12].

Each such color $[R, G, B]$ forms an *equivalence class* [11] in the linear space of possible incident flux distributions $p_i(\lambda)$ (i.e., a class of functions p_r that all give the same color perception). The three cone response functions are the basis for this space, and $[R, G, B]$ is the expression of the color in this basis. Note that this equivalence class is not a linear subspace because of necessary positivity constraints. Indeed, all the quantities in (6.4-1 through 6.4-3) must be nonnegative, and this leads to difficulties in color matching between and among image sensors and displays.

In 1931 the CIE performed colorimetric experiments that defined three color-matching functions for three narrowband primaries centered at $R = 435.8$ nm, $G = 546.1$ nm, and $B = 700$ nm [10]. These measured quantities are plotted in Figure 6.4-2, where negative values indicate that color could not be matched with the three chosen primaries. In these cases, the negative of the indicated value (now positive) was added to the test color in order to make the match. For example, we see in the figure that negative s_R is indicated between about 440 and 550 nm. So if we are trying to match a monochromatic color in this range, we cannot do it with additional R . Instead, we add some R into that monochromatic test color, and in the positive amount $-R$, to make a match. Mathematically, this is equivalent to using a negative amount of primary R [10]. Subsequently, to avoid the negative values in

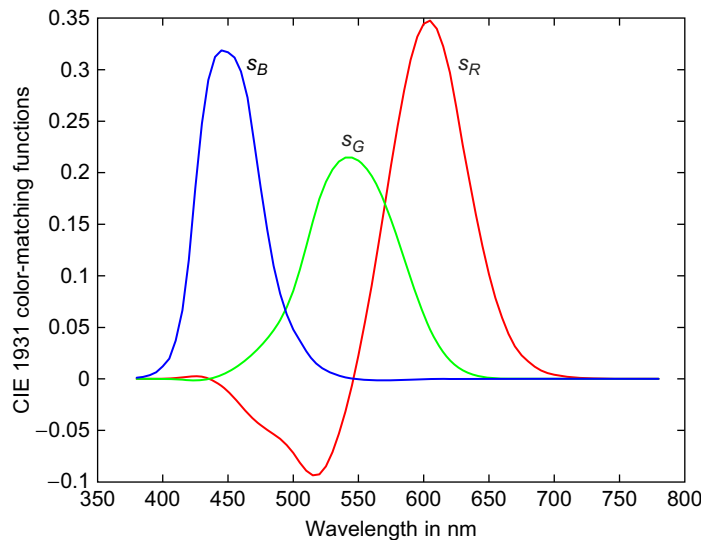
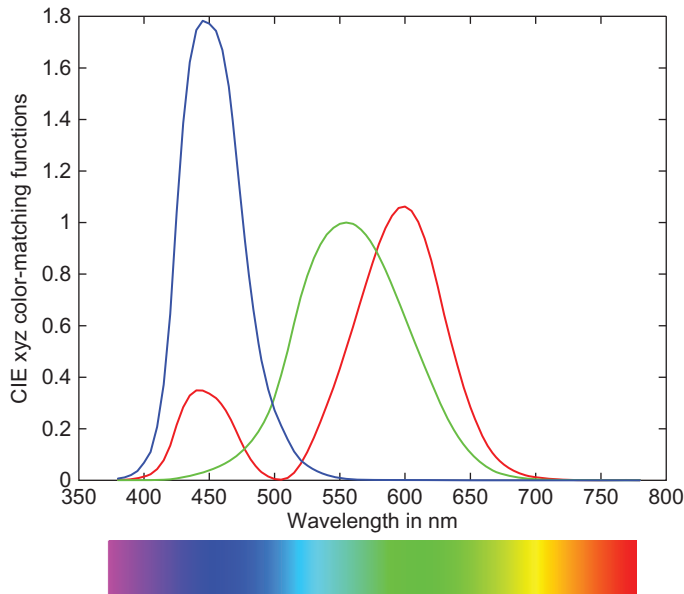


FIGURE 6.4-2

CIE RGB color-matching (color sensitivity) functions (1931).

**FIGURE 6.4-3**

CIE 1931 X, Y, Z color-matching functions (tristimulus values), also called $\bar{x}, \bar{y}, \bar{z}$ functions.

these basis (sensitivity) function curves, the CIE defined a linear transformation on the color-matching functions that avoids the negative values, albeit now with unrealizable primaries, yielding the so-called CIE XYZ color-matching functions [10] that are in wide use to this day. These color-matching functions are plotted in Figure 6.4-3.

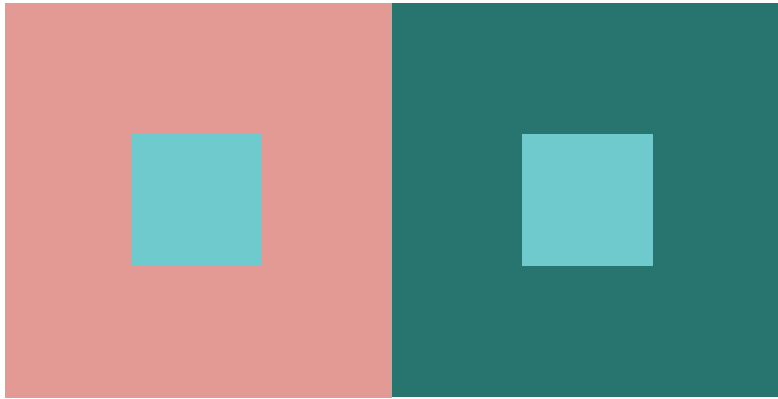
The success of this color matching depends on keeping the background illumination constant. This is because of local *background adaptation*, illustrated in Figure 6.4-4, which shows two equal-colored blocks against light- and dark-colored backgrounds. We can see the block on the left does not seem to exactly match the color of the one on the right. The color background maintained by the CIE in their experiments was an approximation of daylight, which is the 6500° blackbody radiation D65 [10].

Chromaticity Values

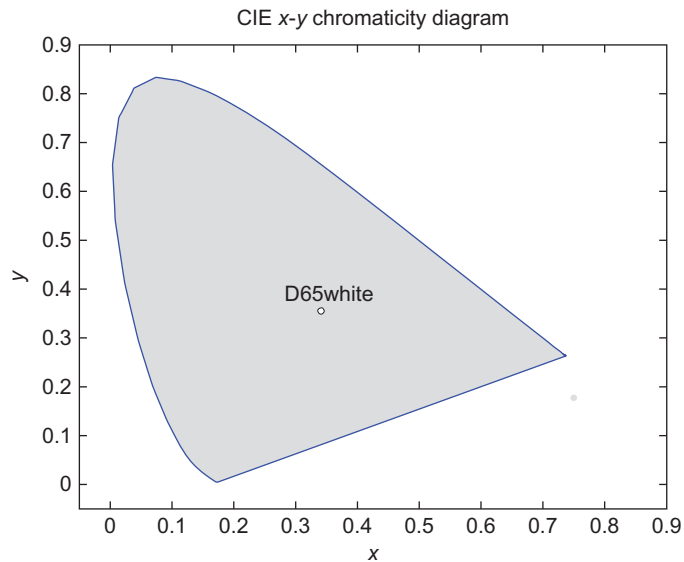
Focusing just on chrominance, and neglecting luminance, we can form a so-called *chromaticity diagram* made from the various tristimulus values by sweeping through wavelength λ and the plotting *chromaticity* (x, y) :

$$x \triangleq X/(X + Y + Z),$$

$$y \triangleq Y/(X + Y + Z).$$

**FIGURE 6.4-4**

The small block has the same color in both sides of the figure.

**FIGURE 6.4-5**

CIE 1931 chromaticity diagram of the standard observer.

The chromaticity diagram is thus a normalized plot of the color response of the CIE 1931 standard observer. Each point on the curved boundary line in [Figure 6.4-5](#) is plotted from the tristimulus values that result from a single-wavelength stimulus. Taking the individual CIE 1931 XYZ color-matching values for each wavelength,

we can make a plot as a hypothesized narrowband source sweeps across the visible wavelength $380 \leq \lambda \leq 780$ nm. The whitepoint is indicated for the D65 standard illuminant available at the CIE Web site [13].

Color Processing

When the human eye is directly exposed to the light intensity, the CIE standard observer provides a good indication of the visual system response, which we quantify as XYZ or CIE RGB tristimulus values, the “color.” But when we place an image sensor and image display device between the object and the observer, things change. First, the typical image sensor has three color channels with their own sensitivity curves, producing sensor R, G, B values R_s, B_s, G_s . These values in turn are input to an image display with typically three output colors with their own intensity curves. The following example illustrates this situation.

Example 6.4–1: Color Image Sensed and Displayed

Consider a single color object that is photographed (sensed) and then displayed with the color spectrum of Figure 6.4–6.

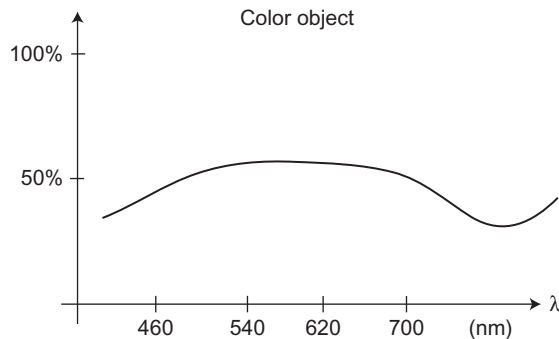


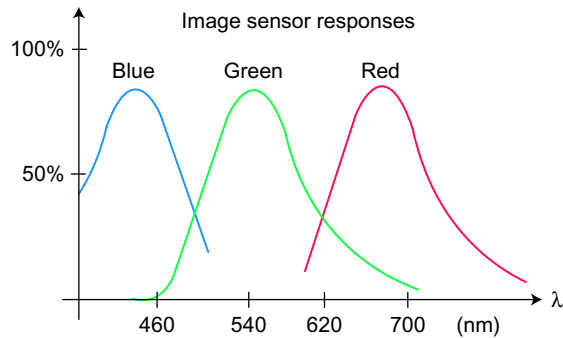
FIGURE 6.4–6

Wavelength spectrum of a color object.

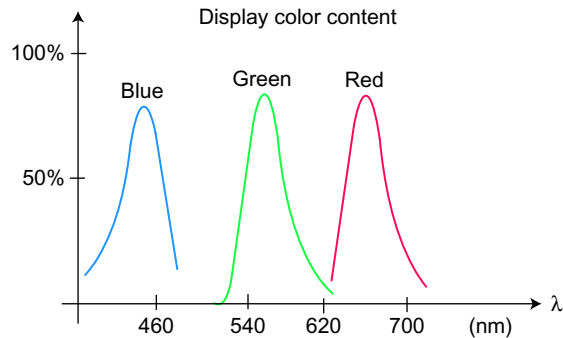
While it is sensed, it is illuminated by a white light (i.e., a flat color spectrum) and recorded with the image sensor’s sensitivity functions shown in Figure 6.4–7.

The resulting sensed R_s, B_s, G_s values are the integrals of the point-by-point product of these two curves [12]. Assuming no signal processing of these values, the display device adds in its sensitivity curves based on its red, green, and blue light sources, with different wavelength curves shown in Figure 6.4–8, and creates the displayed image.

Finally, the output light of the display device excites the HVS sensitivity curves that we model as XYZ. After these various transformations, the human perceived color may be much different than that which would have been perceived on direct viewing of the object.

**FIGURE 6.4-7**

Possible color sensor response functions.

**FIGURE 6.4-8**

Sketch of possible display color primaries.

Since we just have a solid color here, the main question is whether the single displayed color is metameric to the original one. The displayed output will be a weighted combination of the three display curves in Figure 6.4-8, which clearly cannot equal the object spectral response shown in Figure 6.4-6. However, from the viewpoint of our linear XYZ model of the HVS response, we just want the displayed color to be perceived as the same. For this we need only for the integrals in (6.4-1), (6.4-2), and (6.4-3) to be the same, because then, up to an approximation due to this linear model, the cones will output the same signal as when the color object was directly perceived. Clearly, without careful choice of these curves and scaling of their respective strengths, the perceived color will not be the same more often than not. Careful calibration [10, 11] of the color display monitor is necessary.

An approximate solution to this problem is to *white balance* the system. In this process, a pure white object is imaged first and then the system output R , G , and B values are scaled so as to produce a white perception in the HVS of the (standard) viewer. The actual signal captured by the camera may not be white because of possible (probable) nonwhite illumination, but the scaling is to make the output image be perceived as white when viewed in its display environment. ■

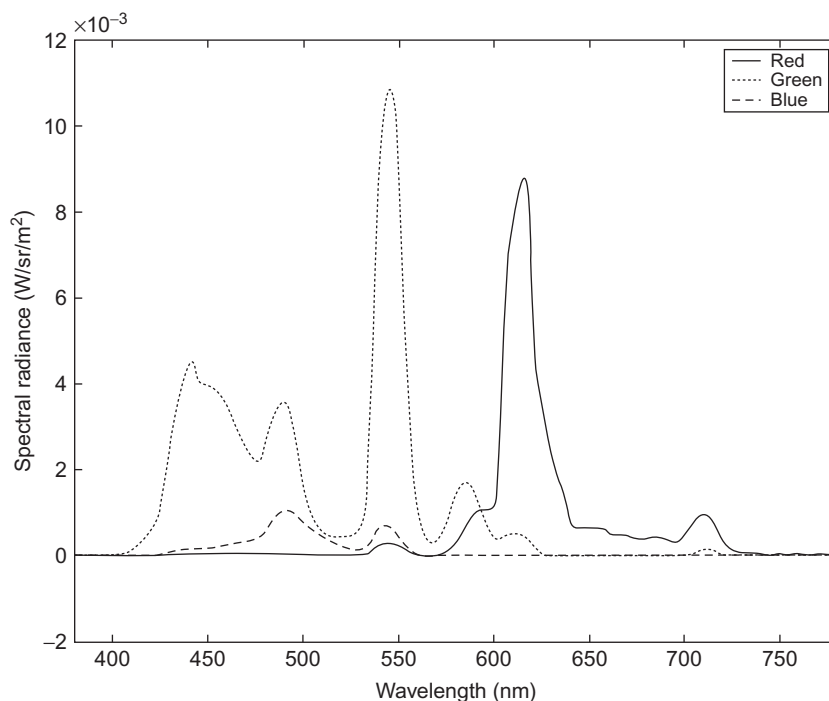


FIGURE 6.4-9

Spectral radiance functions from the *R*, *G*, and *B* channels of an LCD panel. (from [20]
©2002 IEEE)

You might guess that the color display problem would be simpler if the display primaries were monochromatic, but this is usually not possible. Figure 6.4-9 shows an example of the RGB display response functions of an LCD panel [14] circa 2002. We see that red and green are rather impulsive, with blue being decidedly less so.

We can characterize the color spaces of such image input/output devices and relate them to CIE XYZ space. These would be *device-dependent* color spaces and are often proprietary.

6.5 COLOR SPACES

Some *device-independent* color spaces have become useful both for standardization purposes and for evaluation of perceived color uniformity. A few of them are presented here. First, we look at the CIELAB space that targets perceptual uniformity. Then we look at the so-called Rec. 709 color space of high-definition television (HDTV) and the sRGB color space of digital photography and printing. These device-independent color spaces can serve as common reference points when

converting from the device-dependent color space of a scanner or camera to the device-dependent color space of a display or printer. The industry-supported International Color Consortium (ICC) [15] specification permits device color management systems to easily perform such conversions through a well-defined profile connection space.

CIELAB

This color space is a nonlinear transformation on the CIE XYZ space with the purpose of making the space more uniform with respect to human perceived color differences. The CIELAB color coordinates are given as

$$\begin{aligned} L^* &\triangleq 116f(Y/Y_n) - 16, \\ a^* &\triangleq 500[f(X/X_n) - f(Y/Y_n)], \\ b^* &\triangleq 200[f(Y/Y_n) - f(Z/Z_n)]. \end{aligned}$$

Here, the X_n, Y_n, Z_n are the tristimulus values of the reference white under a reference illumination, and f is the pointwise nonlinearity

$$f(x) \triangleq \begin{cases} x^{1/3}, & x > 0.008856, \\ 7.787x + 16/116, & x < 0.008856. \end{cases}$$

The CIELAB space is often used for perceptual comparisons in image science because a small change ($|\Delta L^*|, |\Delta a^*|, |\Delta b^*|$) has been found to have a much more uniform perceptual effect [10] than does the corresponding small change ($|\Delta X|, |\Delta Y|, |\Delta Z|$). This color space can be used to evaluate the color uniformity of a color image display. It is an example of a *nonlinear color space*.

International Telecommunication Union (ITU) Recommendation 709

Another transformation from XYZ is specified in ITU Recommendation ITU-R BT.709 [16] and is given by Poynton [17]. First, a linear transformation is specified as

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

The inverse transformation is given as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950277 \end{bmatrix} \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix}. \quad (6.5-1)$$

These are linear RGB values—i.e., linear in terms of light intensity. Second, the ITU Rec. 709 specification for HDTV puts forth the precise nonlinear function

$$g(x) = \begin{cases} 4.5x, & x \leq 0.018, \\ 1.099x^{0.45} - 0.099, & x > 0.018 \end{cases}$$

to be applied to each linear R_{709} , G_{709} , and B_{709} intensity domain value to produce the more perceptually uniform components R'_{709} , G'_{709} , and B'_{709} , respectively:

$$R'_{709} \triangleq g(R_{709}), G'_{709} \triangleq g(G_{709}), \text{ and } B'_{709} \triangleq g(B_{709}).$$

The inverse of this transform, performed (approximately) at the image display (video monitor), is given as

$$g^{-1}(x') = \begin{cases} x'/4.5, & x' \leq 0.081, \\ \left(\frac{x'+0.099}{1.099}\right)^{1/0.45}, & x' > 0.081. \end{cases}$$

Usually Rec. 709 color space means the nonlinear $R'_{709}, G'_{709}, B'_{709}$ color space, rather than the linear one, $R_{709}, G_{709}, B_{709}$.

A related nonlinear color space is Rec. 709 Y', C_B, C_R , defined via the transformation [18]

$$\begin{aligned} Y' &\triangleq 0.2126R'_{709} + 0.7152G'_{709} + 0.0722B'_{709}, \\ C_B &\triangleq 0.5389(B'_{709} - Y'_{709}), \\ C_R &\triangleq 0.6350(R'_{709} - Y'_{709}). \end{aligned}$$

The advantage of this space is efficiency, since the three $R'_{709}, G'_{709}, B'_{709}$ channels are highly correlated, while the Y', C_B, C_R channels are approximately decorrelated. It is used mainly in image data compression for storage and transmission purposes.

sRGB

In the sRGB color space, first, a matrix transformation [10, 11] is performed on CIE XYZ tristimulus values:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} \triangleq \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

Then the following nonlinear transformation [10, 11] is applied:

$$g(x) \triangleq \begin{cases} 12.92x, & x \leq 0.0031308, \\ 1.055x^{0.41666} - 0.055, & 0.0031308 < x \leq 1. \end{cases}$$

The sRGB color space values then are $R' \triangleq g(R)$, $G' \triangleq g(G)$, and $B' \triangleq g(B)$. This space is often referenced in color-still cameras and printing devices.

Both Rec. 709 and sRGB nonlinear color spaces are much more perceptually uniform than is CIE XYZ but not nearly as uniform as CIELAB. Their main use is as

standard intermediate color spaces for sharing images and video. In this way the camera designer tries to optimize the camera for display in, say, the sRGB space. Then the color printer designer assumes that the input image is expressed in this space and goes on to design the display system to do a good job of conversion between the standard sRGB space and the proprietary color space of that output device. These standard color spaces then serve as an intermediate target for the manufacturers of the various components in the image chain (i.e., camera, processor, coder, decoder, display, and printer). More on color representation and display is contained in Trussell's Chapter 8 of *The Essential Guide to Image Processing* [19].

6.6 IMAGE SENSORS AND DISPLAYS

In this section we will overview both electronic and film sensors, followed by a brief introduction to image and video display devices.

Electronic Sensors

Electronic image sensors today are mainly solid state and of two types: *charge-coupled device* (CCD) and *coupled metal-oxide semiconductor* (CMOS). At present, CCD sensors tend to occupy the niche of higher quality and cost, while CMOS sensors generally have somewhat lower quality but at a much lower cost, having benefited from extensive silicon memory technology. These sensors are present in virtually all current video cameras as well as digital image cameras. They are generally characterized by the number of pixels, today running up to a total of 16M pixels for widely available still-image cameras and 36M for high-end professional still cameras. While still image sensors read out their frames rather slowly (e.g., 1–10 frames per second fps), video sensors must output their image frames at a faster rate (i.e., 24–100 fps), and so their pixel counts tend to be lower. Also, still-image sensors usually have deeper charge wells and so accommodate a wider dynamic range, as measured in f-stops² or bit depth, currently at 10 bits or more. For a digital still camera, access to its 12- to 14-bit image sensor data is usually available only in the camera's `.raw` format that is typically prior to gamma compensation³ and demosaicing. While the format is company proprietary, there are intermediate standards such as `.dng` and `.tiff/ep` with converters widely available on the Internet. When `.raw` data is losslessly compressed, the file size is much larger than typical JPEG or `.jpg` file outputs that are normally gamma compensated, demosaiced, and of 8-bit depth on each color.

²The f-stops on a camera are measures of the aperture, or opening in front of the camera lens. Each increment in f-stop results in a halving of the light intensity on the lens.

³Gamma compensation is a nonlinearity inserted by camera makers to compensate for an assumed nonlinearity ($x^{2.2}$) of display devices. See **Gamma** later in this section.

In common digital still cameras today, there is only one sensor for all three primary colors. A mosaic color filter called *color filter array* (CFA) is applied to the front end of the imaging chip to effectively subsample the three colors. An alternative, often used in video cameras, is a color prism and three imaging chips, one for each of R , G , and B . A novel multilayer CMOS sensor [20] has been developed that stacks R , G , and B pixels vertically and uses the wavelength filtering properties of silicon to avoid the use of the relatively heavy and cumbersome prism. However, it has not been widely accepted. Both of these solutions have an advantage over the CFA sensor in that three-color information is available at every pixel location, rather than only on a checkerboard pattern. Of course, one can just put in more pixels to partially make up for the lost resolution, but that usually means smaller pixels with lessened lowlight capability, if the sensor size is kept constant. Nevertheless, it is the solution of choice for digital still cameras, and the CFA solution has recently moved into the realm of high-end professional video cameras that use single large 35-mm size CMOS sensors.

The most widely used CFA is the Bayer array⁴ [21] shown in Figure 6.6–1. We see that a basic 2×2 cell is repeated to tile the sensor surface, effectively subsampling each of the three color primaries R , G , and B . The Bayer CFA favors green, as does the HVS, and it subsamples red and blue by a larger factor. The actual subsampling of G is 2:1 on a diamond lattice, with R and B subsampled on offset 2×2 Cartesian lattices. Some internal processing in the digital camera, called *demosaicing*, tries to make up for the resulting aliasing, which can be controlled by a slightly defocused

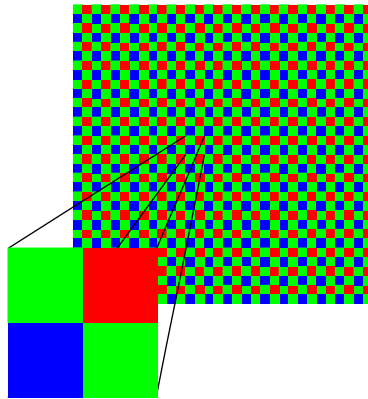


FIGURE 6.6–1

2×2 -pixel cell in Bayer CFA.

⁴Named for Bryce Bayer of Eastman Kodak Company, U.S. Patent No. 3,971,065, *Color Imaging Array*, 1976.

lens design. The in-camera postprocessing generally does a good job, but still, for certain images with much high-frequency color information that is near periodic, aliasing errors are visible [21]. Strictly speaking, the sensors do not really sample, rather they count up photons inside a square or rectangular pixel, called the *aperture effect*. As such, it can better be modeled as a box filter followed by an ideal sampler, thus giving some small degree of lowpass filtering, which tends to suppress spatial frequency alias energy.

The quality of an image sensor is also defined by what its light sensitivity is and how much light it takes to saturate it. A large *dynamic range* for electronic image sensors is difficult to achieve, without a large cell area to collect the light, hence forcing a trade-off with pixel count for a given silicon area. The other variable is *exposure time*. Generally, higher dynamic range can be achieved by lengthening the exposure time, but that can lead to either motion blurring or too low a frame rate for use in a video sensor. Large-format CMOS sensors use a Bayer filter array and have no color prism. Because of their large size, they have deeper electron wells for a given resolution, and hence larger dynamic range. Scientific or sCMOS sensors [22] have been announced with dynamic ranges of 16 bits at 30 fps. These electronic image sensors tend to provide an output that is linear in intensity, having simply counted the photons coming in. Nevertheless, it is common for camera makers to put in a nonlinearity to compensate for the nonlinearity of the previously common *cathode ray tube* (CRT). This is the so-called gamma compensation function. We talk more about gamma in a following section on image displays.

The sensor noise of CCD and CMOS tends to be photon counting related. First, there is a background radiation giving a noise floor, typically assumed to be Poisson distributed. Then the signal itself is simply modeled as a Poisson random variable, with mean equal to the average input intensity, say λ , at that pixel cell. A key characteristic of the Poisson distribution is that the variance is equal to the mean, so that the rms signal-to-noise ratio becomes $\lambda/\sqrt{\lambda} = \sqrt{\lambda}$. This means the SNR is better at high levels than at low levels, and the noise level itself is intensity dependent. For even modest counts, the Poisson distribution can be approximated as Gaussian [23], with mean λ and standard deviation $\sqrt{\lambda}$, or what is the same, variance λ . This means that there is an additive noise in the intensity signal, but rather than being constant variance, it has a variance that depends on the intensity, a so-called intensity-modulated white Gaussian noise.

Sensor Fill Factor

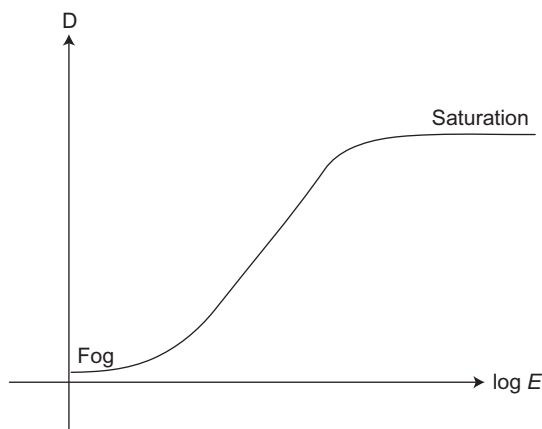
The image sensor usually does not have an active photon collecting area equal to its footprint, giving rise to a ratio called *sensor fill factor*. If the fill factor is large, this helps to reduce the aliasing due to spatially sampling the continuous image field. However, if the fill factor is small, then some kind of anti-alias filtering should be used. It can take the form of the camera lens, maybe slightly defocused, or there can be some spatial lowpass filtering of the sensor output image to reduce the high frequencies where aliasing is most likely to occur.

Some high-resolution image sensors on digital still cameras also function as video sensors with a reduced resolution. The reduced resolution can be obtained in two ways. The preferable way would be to process the full-resolution image with a spatial lowpass filter and then digitally subsample the result down to the lower desired video resolution. An alternative, cheaper, and faster way is just to output the subsampled sensor cells directly, however this can sometimes give rise to serious spatial frequency aliasing error in highly detailed images. To provide some specific numbers for this problem, as of this writing, an image sensor on a widely available digital still camera may have about 4000–5000 pixels horizontally and 3000–4000 pixels vertically. Typical high-definition (HD) video resolutions are around 2000 by 1000 pixels, and standard definition (SD) resolutions are around 700 by 500 pixels. A 3:1 vertical subsampling is commonly employed to achieve HD video frame rates (24–30 fps). Serious spatial frequency aliasing errors have been reported on the video output from such cameras. Camera operators have developed various “workarounds,” all related to defocusing such aliased areas.

Film

The highest resolution and dynamic range image capture has historically been film based. Images are captured on 70-mm film in IMAX cameras at resolutions in excess of $8M \times 6M$ pixels. The spatial resolution of conventional 35-mm movie film is generally thought to be in a range from $3M$ to $4M$ pixels across, depending on whether we measure the camera negative or a so-called *distribution print*, which may be several generations removed from the in-camera negative. Further, film has traditionally excelled in dynamic range—that is, the ability to simultaneously resolve small differences in gray level in both very bright and very dark regions. Because of Weber’s law, the human eye is more sensitive to changes in the dark regions, where film generally has had an advantage over electronic sensors, although this is currently being challenged by the latest solid-state sensors.

Digital images often were initially captured on film in an effort to achieve the highest quality. The film is then scanned by a device that shines white light through it and records the *optical density* of the film. There are various names for such a device, including film scanner, microdensitometer, and telecine. Film is generally characterized by a $D - \log E$ curve, where D is the film’s optical density and E is *exposure*, defined as the time integral of intensity over the exposure time interval. A typical $D - \log E$ curve is sketched in Figure 6.6–2. We see that there is a broad linear range bordered by “saturation” in the bright region and “fog” in the dark region. The linear part of the $D - \log E$ curve gives the useful dynamic range of the film, often expressed in terms of f-stops. Note that density in the linear region of this curve is quite similar to contrast as we have defined it in (6.2–1) and is therefore much more nearly perceptually uniform than is intensity. Digitization of film is thus most often done in the “density domain,” giving us effectively a record of contrast. Since film is completely continuous, there is no spatial frequency aliasing to worry about, at least neglecting its random grain structure.

**FIGURE 6.6–2**

Sketch of the $D - \log E$ curve of film.

Film itself is composed of “grains” that have a probability of being “exposed” when struck by an incoming photon. As such, film-like CCD and CMOS chips also counts photons, both incoming from the imaged object and background radiation. The fog area in Figure 6.6–2 corresponds to film grains exposed by this background radiation. The saturation area in the $D - \log E$ curve corresponds to the situation where most all the grains are already exposed, so a longer exposure time or greater average incoming light intensity will not expose many more grains, and finally the density of the film will not increase any further.

Image and Video Display

Images, once acquired and processed, can be displayed on many devices, both analog and digital. In the “old” technology area, we have photographic prints and slides, both positive and negative. In the classic video area, we have film projectors and CRTs for electronic video, such as television. Modern technology involves electronic projectors such as liquid crystal digital (LCD), liquid crystal on silicon (LCOS), and digital light processing (DLP) and plasma. There are also flat-panel plasma displays made using miniature tubes, as well as the ubiquitous smaller LCD panels used in both stationary and portable computers. Each of these display categories has issues that affect its use for a particular purpose (e.g., display resolution, frame rate, gamma, dynamic range, noise level, color temperature). It can be said that no one display technology is the best choice for all applications, so that careful choice of alternatives is needed.

Common LCD displays are a transmissive technology and use a single cold cathode fluorescent lamp (CCFL) light source behind the screen constituting a so-called *backlight*. It is hard to obtain true black in such a situation because, even in the pixel off-state, some light shines through the LCD panel. Recently, LED light sources have

begun to replace the CCFL backlight due to reduced energy requirements and longer lifetimes. A low-resolution array of LED pixels can be used to provide local dimming behind a high-resolution LCD array for the purpose of increased perceived dynamic range. Commonly, though, the LEDs are only arranged on the perimeter of the LCD display, and this does not give any increase in perceived dynamic range; it can, however, make the display thinner and more energy efficient.

Flicker was a problem with the CRT because the pixel was only illuminated instantaneously followed by a rapid phosphor decay for the rest of the display cycle. For solid-state displays, though, due to their constant pixel illumination, flicker is not a problem. However, there is a new problem with *judder*, the discontinuous or stepwise motion that can be perceived from these displays. While judder had been seen in movie projectors, it is relatively unknown in CRT displays.

LCD displays are ubiquitous as computer monitors and image projectors offer good spatial resolution and color rendition. They have had a problem with transient response, which tends to leave a trail on the screen from moving objects, but this has been largely addressed by 120 fps models (also called 120 Hz).

Plasma panels are very bright, and the technology can scale to large sizes, but there can be visible noise in the dark regions due to the pulse-width modulation system used in their gray-level display method. In a bright room, this is not usually a problem. They tend to have the best dark levels, though, better than LCDs, which use a transmissive technology.

In the projector area, very high spatial resolutions have been achieved by a variant of LCOS in a 4K digital cinema projector. Another projection technology is DLP chips that contain micromirrors at each pixel location. Each micromirror is tilted under control of each digital pixel value to vary the amount of light thrown on the screen for that pixel. It can offer both excellent black levels and transient response in a properly designed projector. DLPs have also achieved 4K resolution, meaning a resolution of 4K pixels horizontally and 2K or more vertically.

The CRT is still the only common device that directly displays interlaced video—i.e., without deinterlacing it first.⁵ Since deinterlacing is not perfect, this is an advantage for an interlaced source such as common SD video and HD 1080i. Today, however, much program origination is done by progressive film and video cameras (720p and 1080p) that well suit the now-common LCD and plasma displays.

Gamma

Until recently, the standard of digital image display was the CRT. These devices have a nonlinear display characteristic, usually parameterized by γ as follows:

$$I = v_{\text{in}}^{\gamma},$$

where v_{in} is the CRT input voltage, I is the light intensity output, and the gamma value γ usually ranges from 1.8 to 2.5. Because of this nonlinearity, and because of

⁵Deinterlacing is covered in Chapter 10.

the widespread use of CRTs in the past, video camera manufacturers have routinely implemented a *gamma correction* into their cameras for nominal value $\gamma = 2.2$, thus precompensating the camera output voltage as

$$v_{\text{out}} = v_{\text{in}}^{1/\gamma}.$$

Since v_{in} is proportional to I for a CCD or CMOS image sensor, this near square-root function behaves somewhat similar to the log used in defining contrast, and so the usual camera output is more nearly contrast than intensity. Modern flat panel display devices are near linear and don't need gamma compensation, however current models implement the gamma nonlinearity in their processing chain.

Notice that film density has a useful response region, shown in Figure 6.6–2 that is linear in exposure E . For constant intensity I , over a time interval T , we then have approximately, and in the linear region,

$$d = \gamma \log I + d_0,$$

so that film density is contrast-like also and thereby, via Weber's law, obtains the advantage of a more uniform perceptual space.

High Dynamic Range and Tone Mapping

The dynamic range of the human eye is quite large. Also the range of light intensities seen in nature is much larger than that. Quantizing in density, we typically use 8 bits for digital images and achieve only a window on what is possible. The term high dynamic range (HDR) means imaging with more than 8 bits in contrast (density). When images are viewed in a bright room, usually an HDR display is not required; however, common viewing conditions for movie theaters involve darkened rooms where the HDR range of human vision can be more fully addressed. For a full HDR system, we need HDR cameras (sensors) as well as HDR displays/projectors. When the full captured dynamic range cannot be displayed, a pointwise nonlinear function, or *tone mapping*, is used to reduce the dynamic range to fit the display and viewing environment.

One way to acquire HDR images with a conventional camera is to take short and long exposures in rapid succession. Some commonly available digital still and smart phone cameras take this approach, tone mapping the result before storing in JPEG format. Access to camera .raw files can permit higher dynamic range output from a single exposure.

In the output area, and as mentioned earlier, HDR display is possible with LED low-resolution backlight sandwiched with an LCD transmissive display [24]. This way, the local brightness can be modulated to achieve higher dynamic range. In one implementation, the LED array would be given the upper 8 bits of a local image value, while the LCD array would show the detail in the lower 8 bits of the pixel value. An HDR display based on this approach was introduced in 2005 [25], but has not become widely available.

CONCLUSIONS

The basic properties of images, sensors, displays, and the human visual system (HVS) studied in this chapter are essential topics, providing a link between the multi-dimensional signal processing of Chapters 1–5 and the image and video processing of the following chapters. In the coming chapter, we will make use of several of these properties for both image/video processing and compression.

PROBLEMS

- One issue with the $YC_R C_B$ color space comes from the constraint of positivity for the corresponding RGB values.
 - Using 10-bit values, the range of RGB data is $[0, 1023]$. What is the resulting range for each of Y , C_R , and C_B ?
 - Does every point in the range you found in part (a) correspond to a valid RGB value—i.e., a value in the range $[0, 1023]$?
- From the Mannos and Sakrison human visual system (HVS) response function shown in Figure 6.2–5 that is normalized to 1, how many pixels should there be vertically on a 100-inch vertical screen when viewed at a distance of $3H$ (i.e., 300 inches)? At $6H$? Assume we want the HVS response to be down to 0.01 at the Nyquist frequency.
- Assume the measured JND is $\Delta I = 1$ at $I = 50$. Hence,

$$\frac{\Delta I}{I} = 0.02.$$

Consider a range of intensities $I = I_0 = 1$ to $I = 100$.

- If we quantize I with a fixed step size, what should this step size be so that the quantization will not be noticeable? How many steps will there be to cover the full range of intensities with this fixed step size? How many bits will this require in a fixed length indexing of the quantizer outputs?
- If we instead quantize contrast

$$C = \ln(I/I_0),$$

with a fixed step-size quantizer, what should ΔC be? How many steps are needed? How many bits for fixed-length indexing of these values?

- Referring to the three-channel theory of color vision (6.4–1)–(6.4–3), assume three partially overlapping human visual response curves (luminous efficiencies) $s_R(\lambda)$, $s_G(\lambda)$, and $s_B(\lambda)$ are given. Let there be a camera with red, green, and blue sensors with response functions $s_{cR}(\lambda)$, $s_{cG}(\lambda)$, and $s_{cB}(\lambda)$, where the subscript c indicates “camera.” Let the results captured by the camera be displayed with three very narrow-wavelength beams, here modeled as monochromatic $s(\lambda) = \delta(\lambda)$,

additively superimposed, and centered at the wavelength peak of each of the camera luminous efficiencies. Assuming white incident light (i.e., uniform intensity at all wavelengths), what conditions are necessary so that we perceive the same R , G , and B sensation as if we viewed the scene directly? Neglect any nonlinear effects.

5. In “white balancing” a camera, we capture the image of a white card in the ambient light, yielding three R , G , and B values. We might like these three values to be equal; however, because of the ambient light not being known, they may not be. How should we modify these R , G , and B values to “balance” the camera in this light? If the ambient light source changes, should we white balance the camera again? Why?
6. Consider the “sampling” that a CCD or CMOS image sensor does, as reflected in the input Fourier transform $X(\Omega_1, \Omega_2)$. Assume the sensor is of infinite size and that its pixel size is uniform and square with size $T \times T$. Assume that, for each pixel, the incoming light intensity (monochromatic) is integrated over the square cell and that the sample value becomes this integral for each pixel (cell).
 - (a) Express the Fourier transform of the resulting sample values; call it $X_{\text{sensor}}(\omega_1, \omega_2)$ in terms of the continuous Fourier transform $X(\Omega_1, \Omega_2)$.
 - (b) Assuming spatial frequency aliasing is not a problem, find the resulting discrete-space transform $X_{\text{sensor}}(\omega_1, \omega_2)$ by specializing your result from part (a).
7. Use MATLAB to compute the whitepoint of the CIE chromaticity diagram of Figure 6.4–5 using the standard D65 illuminant available for download at the CIE Web site [13]. Find and plot the R , G , and B points on the chromaticity diagram corresponding to $R = 435.8$ nm, $G = 546.1$ nm, and $B = 700$ nm.
8. Use the column vectors in the linear color space transformation in (6.5–1) to compute X, Y, Z values of the Rec. 709 spectral primaries. Then calculate the corresponding x, y values and plot them on the chromaticity diagram of Figure 6.4–5. Finally, argue that the color gamut of Rec. 709 color space is a triangle with these three points at vertices.
9. Redo problem 8 for the sRGB color space.

REFERENCES

- [1] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [2] J. E. Hardis, “100 Years of Photometry and Radiometry,” *Proc. SPIE*, vol. 4450, 2001.
- [3] P. G. J. Barten, *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality*, PhD thesis, Technical Univ. of Eindhoven, 1999.
- [4] F. L. van Ness and M. A. Bouman, “Spatial Modulation Transfer Function of the Human Eye,” *J. Optical Soc. of America (JOSA)*, vol. 57, pp. 401–406, 1967.

- [5] J. L. Mannos and D. J. Sakrison, "The Effects of a Visual Error Criteria on the Encoding of Images," *IEEE Trans. Information Theory*, vol. IT-20, pp. 525–536, July 1974.
- [6] D. H. Kelly, "Motion and Vision, II Stabilized Spatio-temporal Threshold Surface," *Journal of Optical Soc. America*, vol. 69, pp. 1340–1349, 1979.
- [7] D. H. Kelly, "Theory of Flicker and Transient Responses, II Counterphase Gratings," *Journal of Optical Soc. America*, vol. 61, pp. 632–640, 1971.
- [8] C. J. vd. B. Lambrecht and M. Kunt, "Characterization of Human Visual Sensitivity for Video Imaging Applications," *Signal Processing*, vol. 67, pp. 255–269, June 1998.
- [9] A. C. Bovik, Ed., *Handbook of Image and Video Processing*, 2nd Ed., Elsevier Academic Press, Burlington, MA, 2005.
- [10] H. J. Trussell and M. J. Vrhel, *Fundamentals of Digital Imaging*, Cambridge University Press, Cambridge, UK, 2008.
- [11] E. Dubois, *The Structure and Properties of Color Spaces and the Representation of Color Images*, Morgan & Claypool Publishers, 2010.
- [12] E. J. Giorgianni and T. E. Madden, *Digital Color Management: Encoding Solutions*, Addison-Wesley, Reading, MA, 1998.
- [13] CIE (2010–2011). *Downloads*. Available at <http://www.cie.co.at/index.php/DOWNLOADS>. Also available at Web site of Colour and Vision Research Lab, UCL: <http://cvrl.ucl.ac.uk/>
- [14] G. Sharma, "LCDs versus CRTs—Color Calibration and Gamut Consideration," *Proc. IEEE*, vol. 90, no. 4, pp. 605–622, April 2002.
- [15] International Color Consortium. Available at <http://www.color.org/index.xalter>
- [16] ITU, Recommendation ITU-R BT.709, Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange (1990), Geneva: ITU, 1990. [formerly CCIR Rec. 709].
- [17] C. Poynton, *A Guided Tour of Color Space*. Available at http://www.poynton.com/papers/Guided_tour/abstract.html
- [18] Wikipedia (2010). *YCbCr*. Available at <http://en.wikipedia.org/wiki/YCbCr>
- [19] A. C. Bovik, Ed., *The Essential Guide to Image Processing*, 2nd Ed., Elsevier Academic Press, Burlington, MA, 2009.
- [20] Foveon (2010). Available at <http://www.foveon.com>
- [21] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demo-saicking: Color Filter Array Interpolation," *IEEE Signal Process. Magazine*, vol. 22, pp. 44–54, January 2005.
- [22] Andor Technology (2011). *sCMOS Camera*. Available at http://www.andor.com/scientific_cameras/neo_scmos_camera/
- [23] H. Stark and J. W. Woods, *Probability and Random Processes with Applications to Signal Processing*, 3rd Ed., Prentice-Hall, Upper Saddle River, NJ, 2002.
- [24] H. Seetzen, W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, and A. Vorozcovs, "High Dynamic Range Display Systems," *ACM Transactions on Graphics*, 2004.
- [25] Wikipedia (2008). DR37-P. Available at <http://en.wikipedia.org/wiki/DR37-P>

Image Enhancement and Analysis

In this chapter we begin by looking at some simple image processing filters. Then we will use some of them for *image enhancement*, generally taken to mean improving the image in some sense, such as less noise, sharp edges, good contrast, etc. Then we move on to the problem of *image analysis*, meaning the extraction of structure and other useful information from an image. We will first look at detection of lines and edges and then briefly discuss the edge-linking problem that occurs when the located edge pixels on an object boundary have gaps. We then raise our level of analysis to objects and look at problems of *image segmentation*. Simple manual thresholding is followed by the more powerful K-means algorithm. Segmented regions may be disconnected, so we also look at *region growing*, a segmentation technique that uses pixel location giving connected regions. Finally, we will briefly consider the problem of detecting objects in an image.

7.1 SIMPLE IMAGE PROCESSING FILTERS

Here, we introduce a few basic image processing filters, some of which we will use in later chapters, but all of these simple filters are used extensively in the practice of image and video signal processing. These filters and more, with extensive processed image examples, can be found in the image processing text by Gonzales and Woods [1].¹

Box Filter

Perhaps the simplest of the basic image processing filters is the *box filter* used to perform linear averages over image regions. The most common size of box filter is 3×3 , giving the finite impulse response (FIR) impulse response $h(n_1, n_2)$ with mask

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

¹Richard Woods (no relation).

where the n_2 axis is directed downward and the n_1 axis is directed across. In order to make the lowpass filter zero-phase, we place the point $(n_1, n_2)^T = \mathbf{0}$ at the center of the matrix. Box filters come in all square and rectangular sizes, with the scaling constant adjusted to create the average of these pixel values. From Chapter 1, the Fourier transform or frequency response of the $L_1 \times L_2$ box filter is given as

$$H(\omega_1, \omega_2) = \frac{1}{L_1 L_2} \frac{\sin \omega_1 L_1 / 2}{\sin \omega_1 / 2} \frac{\sin \omega_2 L_2 / 2}{\sin \omega_2 / 2},$$

which is not a very good lowpass filter due to its large side lobes. Figure 7.1–1 shows the DFT magnitude plot of a 3×3 box filter (see `box.m` in folder M-files on the book's Web site).

However, a box filter only requires $L_1 L_2$ additions and one multiply (divide) in a straight-forward implementation and so is often used. Problem 1 at the end of this chapter leads you to show how this filter can be implemented recursively to greatly reduce the number of adds.

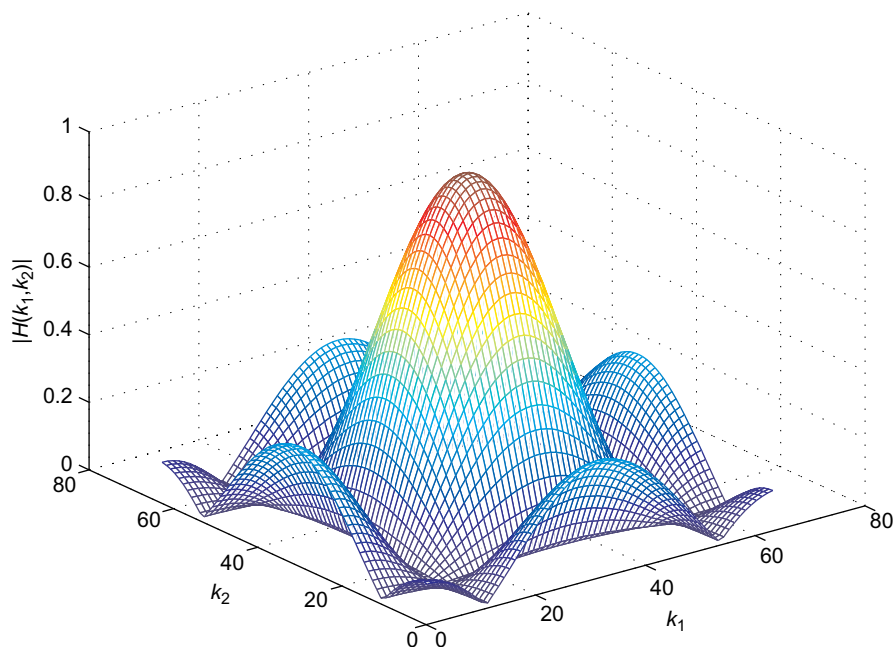


FIGURE 7.1–1

Magnitude frequency response of a box filter obtained with 64×64 DFT in MATLAB.

Gaussian Filter

A simple Gaussian lowpass filter is given in terms of its impulse response as

$$h(n_1, n_2) = \exp -\frac{1}{2\sigma^2} (n_1^2 + n_2^2),$$

with approximate frequency response, over $[-\pi, +\pi]^2$,

$$H(\omega_1, \omega_2) = 2\pi \exp\left(-\frac{1}{2}\sigma^2(\omega_1^2 + \omega_2^2)\right),$$

when $\sigma\pi \gg 1$. Since this impulse response has infinite support, it is truncated to a finite $L \times L$ window centered on the origin. Usually the value of L must be large compared to the box filter, but for this price we are guaranteed no ringing in the lowpass output. A small 5×5 filter with approximate Gaussian shape has been suggested by Burt and Adelson [2]. The filter is separable $h(n_1, n_2) = h(n_1)h(n_2)$, with

$$h(n) = \begin{cases} 0.4, & n = 0, \\ 0.25, & n = \pm 1, \\ 0.05, & n = \pm 2. \end{cases}$$

This filter has been extensively used for image smoothing to reduce both high-frequency content and noise. It can also be applied recursively to create the so-called *Gaussian pyramid* [2, 3].

Prewitt Operator

The Prewitt operator is an unscaled, first-order approximation to the gradient of a supposed underlying continuous-space function $x(t_1, t_2)$ computed as

$$\frac{\partial x}{\partial t_1} \approx \frac{x(t_1 + \Delta t_1, t_2) - x(t_1 - \Delta t_1, t_2)}{2\Delta t_1},$$

but given by a 3×3 FIR impulse response $h(n_1, n_2)$ with mask²

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},$$

again centered on the origin to eliminate delay. Note that the Prewitt operator averages three first-order approximations to the gradient, from the three closest scan lines of the image to reduce noise effects. The corresponding approximation of the vertical gradient,

$$\frac{\partial x}{\partial t_2} \approx \frac{x(t_1, t_2 + \Delta t_2) - x(t_1, t_2 - \Delta t_2)}{2\Delta t_2},$$

is given by the 3×3 FIR impulse response $h(n_1, n_2)$ with mask

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

again centered on the origin to eliminate delay, and again with the n_2 axis directed downwards and the n_1 axis directed across.

²Note this does *not* match matrix notation. To do that would require a matrix transpose of this array.

Sobel Operator

The Sobel operator is a slight variation on the Prewitt operator, with horizontal and vertical masks given as

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix},$$

with the central row being weighted up to achieve some emphasis on the current row or column.

Both the Sobel and Prewitt operators are used widely in image analysis [4] to help locate edges in images. Location of edges in images can be a first step in image understanding and object segmentation, where the output of these “edge detectors” must be followed by some kind of regularization—i.e., smoothing, thinning, and gap filling [1]. A plot of the imaginary part of the frequency response of the Sobel operator, centered on zero, is shown in Figure 7.1–2, with a contour plot in Figure 7.1–3. We can see that the plots approximate a scaled version of the frequency function ω_1 , but only at low frequencies. In particular, the side lobes at high ω_2 indicate that this filter should only be used on lowpass data, or alternatively be used in combination with a suitable lowpass filter to reduce these side lobes.

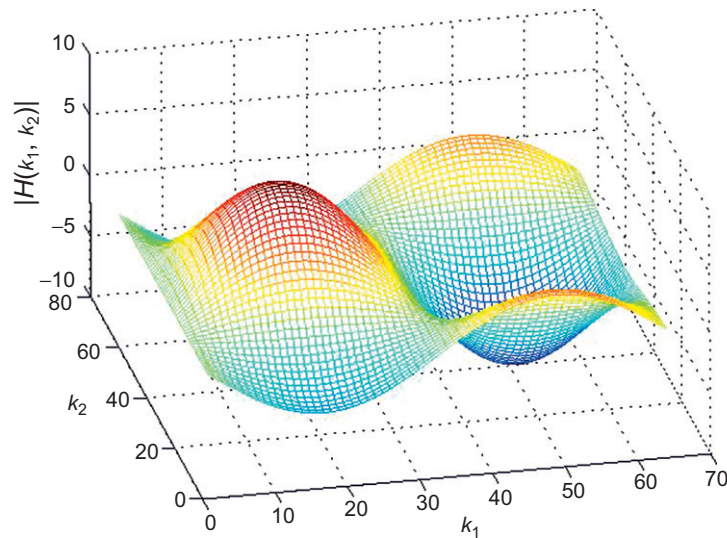
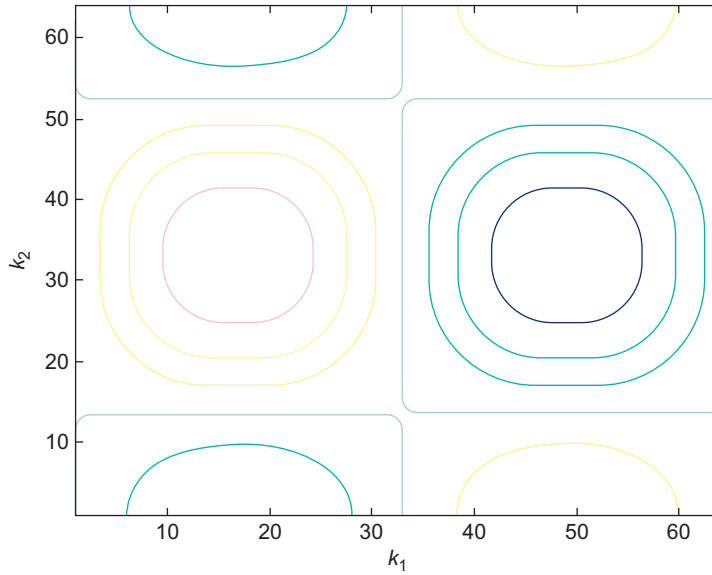


FIGURE 7.1–2

Imaginary part of horizontal Sobel operator frequency response, obtained with 64×64 DFT.

**FIGURE 7.1–3**

Contour plot of the imaginary part of horizontal Sobel operator, obtained with 64×64 DFT.

Laplacian Filter

In image processing, the name *Laplacian filter* often refers to the simple 3×3 FIR filter

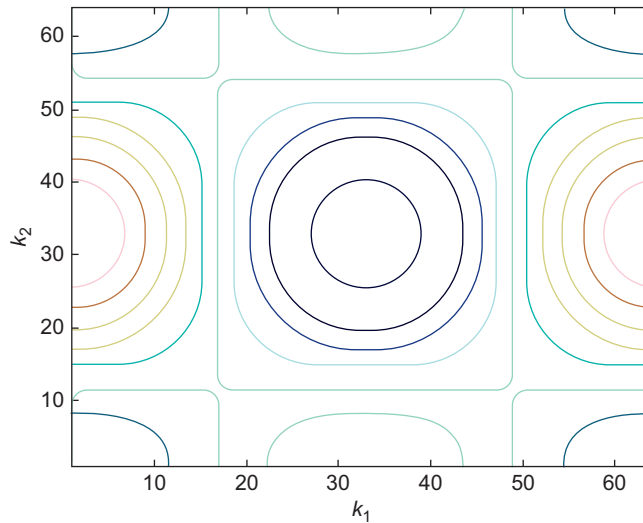
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

used as a first-order approximation to the Laplacian of an assumed underlying continuous-space function $x(t_1, t_2)$:

$$\nabla^2 x(t_1, t_2) = \frac{\partial^2 x(t_1, t_2)}{\partial^2 t_1} + \frac{\partial^2 x(t_1, t_2)}{\partial^2 t_2}.$$

The magnitude of the frequency response of this Laplacian filter is shown in [Figure 7.1–4](#), where again we note the reasonable approximation at low frequencies only.

The zero-crossing property of the Laplacian filter can be used for edge location. Often these derivative filters are applied to a smoothed function to avoid problems with image noise amplification [1]. Another Laplacian approximation is available using the Burt and Adelson Gaussian filter [2].

**FIGURE 7.1–4**

Contour plot of the magnitude frequency response of the Laplacian filter.

7.2 IMAGE ENHANCEMENT

The goal of *image enhancement* is to improve an image in some manner, possibly by reducing the visual noise content or sharpening a blurred image. In this section we will look at two approaches, the first using linear filters and the second using a nonlinear one known as the median filter.

Linear Filtering

We assume a 2-D FIR filter h of size $M \times M$. For the case of additive noise distortion, we take the noise w to be independent and Gaussian $\sim N(0, \sigma^2)$, so that the noisy image is written as

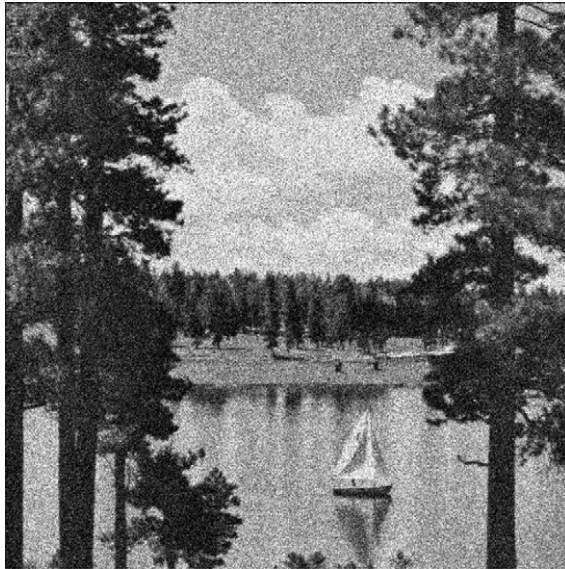
$$y(n_1, n_2) = x(n_1, n_2) + w(n_1, n_2).$$

The noisy *sailboat* image is shown in Figure 7.2–1 for additive noise standard deviation $\sigma = 32$ (variance $\sigma^2 = 1024$), giving a mean square error (MSE) of 1024 with respect to the noise-free image x .

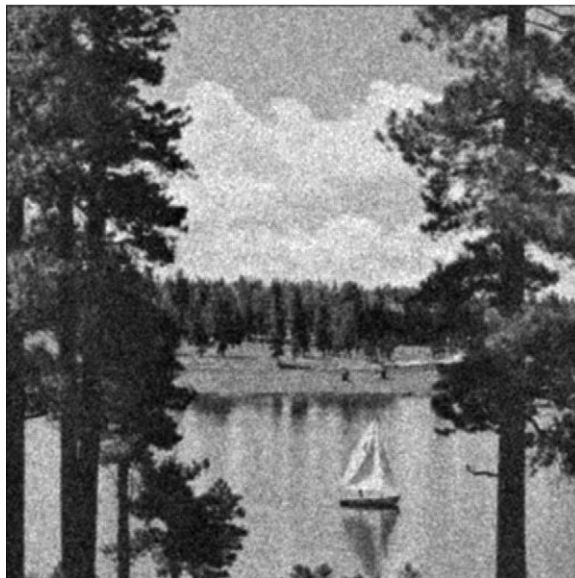
To enhance this noisy image, we consider using a 2-D FIR filter h of size $M \times M$ to produce the estimate \hat{x} as

$$\hat{x}(n_1, n_2) = (h * y)(n_1, n_2).$$

Using the 3×3 box filter mentioned previously, we get the result shown in Figure 7.2–2 with an MSE of 183. We see a smoothing effect on the image and a reduction in the noise. After two passes through the box filter, we get the image in

**FIGURE 7.2-1**

Noisy sailboat (512×512) image with noise variance of 1024.

**FIGURE 7.2-2**

Output after 3×3 box filtering (filtered once).

**FIGURE 7.2-3**

Output after two passes of 3×3 box filter.

Figure 7.2-3 showing more smoothing of the image structures along with a further reduction in the noise and an MSE of 165. Note that repeated convolution with a box filter will eventually result in an output very close to that of a finitesupport approximation to a separable Gaussian kernel.

One way of thinking about these filters is that we are obtaining an approximation to the sample mean for the local data in its convolution window. Since the sample mean minimizes the least-squares error [5], this approach is mathematically motivated.

Filtering in Intensity Domain

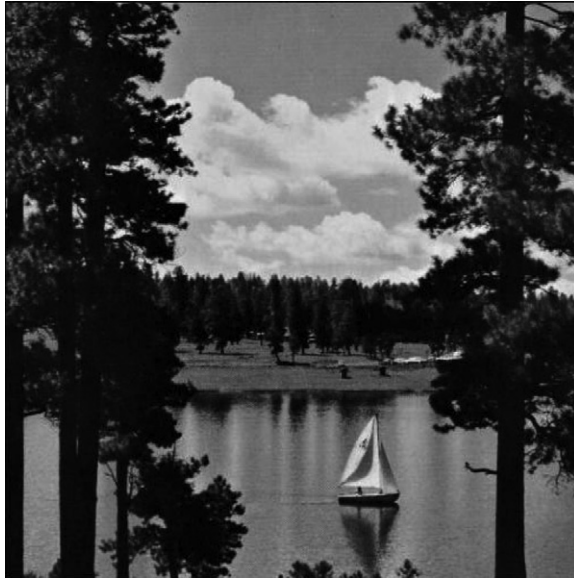
Images are normally available after gamma compensation, what we loosely called here the density domain. If we convert the gamma-compensated images (cf. **Gamma** in Section 6.6) to an approximate *intensity-domain* image via

$$x_{\text{int}} = x^{2.2}, \quad (7.2-1)$$

we get the image shown in Figure 7.2-4. This image has too much contrast because of the gamma nonlinearity that is part of the image display. However, we can try filtering in this intensity domain and then convert back to density for purposes of display. In that case, care must be taken since the noisy image y may have negative values after we add the noise; thus we implement

$$y_{\text{int}} = \max(y_{\text{in}}, 0)^{2.2}.$$

Performing box filtering on the intensity-domain image y_{int} and then converting back to the *density domain* via the inverse of (7.2-1), we get the image shown in

**FIGURE 7.2-4**

Intensity domain image scaled for display.

Figure 7.2-5 after a single pass with an MSE of 267. The corresponding two-pass output is then shown in Figure 7.2-6.

In general, linear filtering of images can be performed in either domain, or in fact, on any nonlinear transformation of the image sensor data. Linear filtering in the intensity domain is analogous to the filtering that can be performed using lenses via the theory of Fourier optics [6]. But linear filtering in the density or contrast domain may be thought more appropriate due to the often additive nature of the approximate Gaussian nature of the noise there (e.g., photon noise and film grain noise). Still, and as pointed out in Chapter 6, this additive noise will tend to have a signal-modulated variance and therefore will not be strictly independent of the signal.

Often the actual nonlinear mapping used in gamma compensation is not known, so a conversion back to intensity is not possible, and maybe not even advisable. For this reason, most of the literature on image noise reduction treats filtering directly upon the given data. This may be starting to change with the advent of so-called `.raw` digital camera output, which often is the full range, say, 16-bit intensity data.

Median Filtering

In addition to the mean, the median is often useful in simple image enhancement. While the mean is the least-squares estimator of a data set, the median is the estimate that minimizes the mean absolute error (MAE) [7]. The simple median filter computes the median or middle number of the pixel values in a sliding square

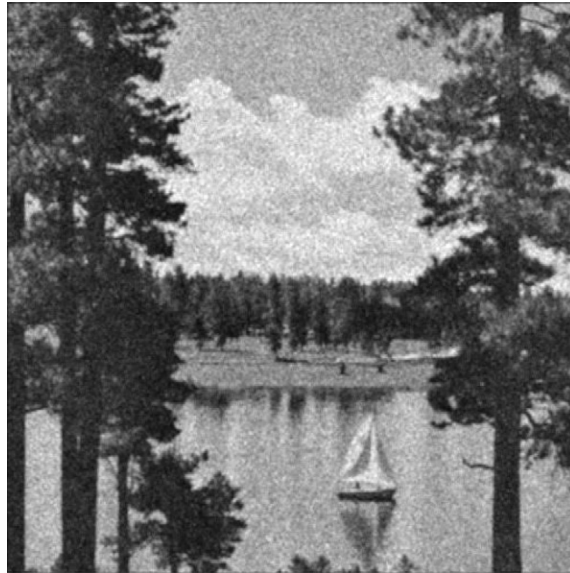


FIGURE 7.2-5

Result of 3×3 box filtering in intensity domain.



FIGURE 7.2-6

Result of two passes through the box filter in intensity domain.

**FIGURE 7.2–7**

Output of 3×3 median filter applied once.

window. It is most easily seen in terms of an ordering operation followed by output of the midvalue. Such nonlinear filters have been found to be much more difficult to analyze than their linear counterparts. However, they produce significantly less blurring of image features and produce good results for certain types of degrading noises.

The image in [Figure 7.2–7](#) shows the effect of 3×3 median filtering on the noisy image in [Figure 7.2–1](#), with resulting MSE of 238. We see that the visual effect of the median filter is similar to that of the linear box filter, although the box filter has done the better job of suppressing the Gaussian noise. If the noise distribution were very different, though, there can be a radical difference. For example, using so-called “salt and pepper” noise—which means that each pixel is independently set to zero with a certain probability p , here, $p = 0.15$ —we get the noisy image shown in [Figure 7.2–8](#). Such a noise is often used to approximately model the effect of direct transmission of a digital image over a binary symmetric channel [8, Sect. 7.4.3]. The median filtered result is seen in [Figure 7.2–9](#). These results were obtained using MATLAB routines `SailboatFiltxxx.m` and at the book’s Web site.

We see that the median filter has almost removed all the salt and pepper noise, and has done this without much visible blurring of edges. For this type of noise, the linear box filter would not do nearly as well. More on nonlinear image filters, including the median filter, can be found in Chapter 12 of *The Essential Guide to Image Processing* [8].

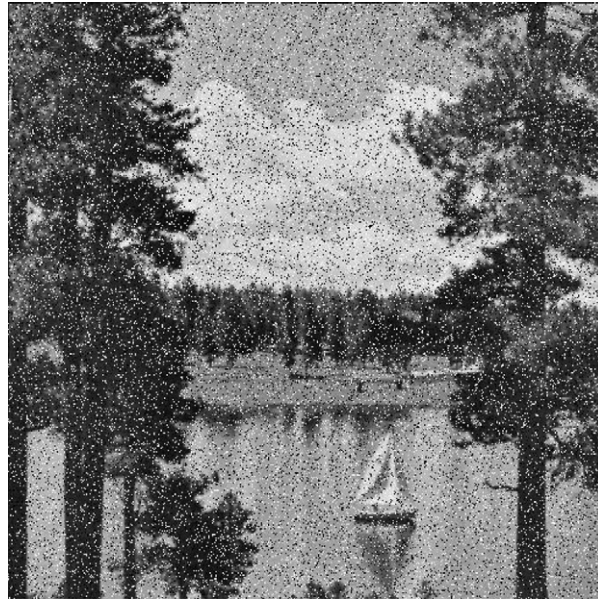


FIGURE 7.2-8

Sailboat with salt and pepper noise, with $p = 0.15$.

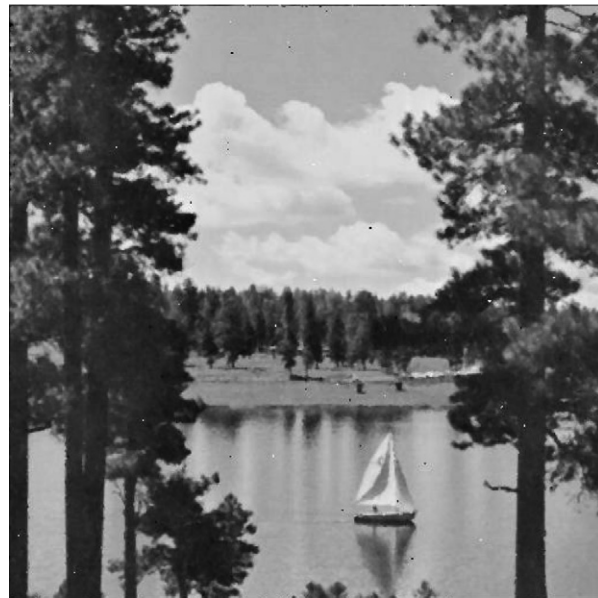


FIGURE 7.2-9

Output of 3×3 median filter of the salt and pepper noisy sailboat (filtered once).

7.3 IMAGE ANALYSIS

Image analysis differs from image enhancement in that the desire is not to produce an “improved” output image, but rather to analyze or understand the image in some way and provide as an output a description of the found structure. An example would be to find the object edges that are very important in human recognition and also serve as building blocks for objects. Other examples could include local means and variances, location of significant feature points and shapes, etc. The resulting increased knowledge of the structure of the image can then aid in computer vision and robotic applications as well as in some advanced image processing methods that take advantage of the image structure.

Edge Detection

The Prewitt and Sobel operators, and other differential operators, can be used to find edges in images. First the image is filtered with, say, the Sobel operator. Then the absolute value of the filter output y is compared to a predetermined fixed threshold γ , and a decision is made. We define the logical variable $\text{Edge}(n_1, n_2)$ to denote the detection of an edge at location (n_1, n_2) as

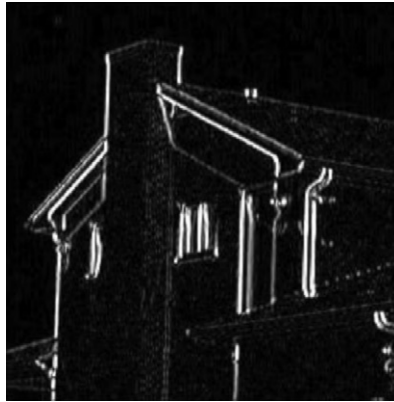
$$\text{Edge}(n_1, n_2) \triangleq \begin{cases} 1, & |y|(n_1, n_2) \geq \gamma, \\ 0, & |y|(n_1, n_2) < \gamma. \end{cases}$$

The logical image $\text{Edge}(n_1, n_2)$ then shows the found locations of edges in the input image y . Consider the *house* image in Figure 7.3–1. If we use the horizontal Sobel operator, we get the absolute value output image in Figure 7.3–2 and the horizontal *Edge* image in Figure 7.3–3 for threshold $\gamma = 200$. The same can be done with the vertical Sobel operator, resulting in the vertical *Edge* image seen in Figure 7.3–4.



FIGURE 7.3–1

256 × 256 gray-level version of the house image.

**FIGURE 7.3-2**

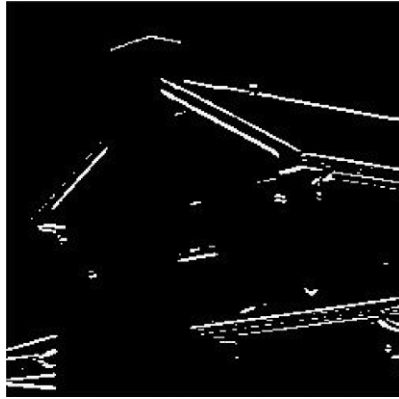
Absolute value of output of the horizontal Sobel filter.

**FIGURE 7.3-3**

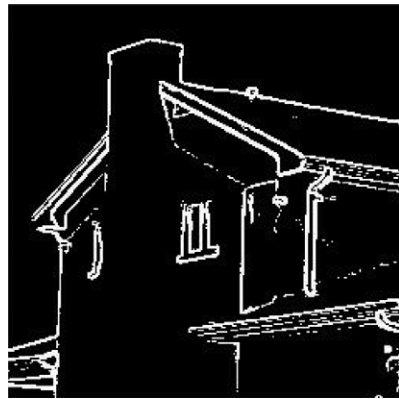
Thresholded horizontal Sobel filter output at $\gamma = 200$.

If we raised the threshold, we would get fewer edge pixels, and the gaps with “missing” edge pixels would get larger. On the other hand, the edge would get “thinner” and more precise. By lowering the threshold, we could get fewer gaps, but the “edges” would get “wider.” Combining the horizontal and vertical absolute values $y = y_v + y_r$, and then thresholding, we get an Edge image that is a combination of horizontal and vertical edges, as seen in [Figure 7.3-5](#) with threshold value $\gamma = 200$.

These images were obtained using the short MATLAB programs `HouseSobVerEdge`, `HouseSobHorEdge`, and `HouseSobEdge` at this book’s Web site. Note that while the combined image contains good edge pixels, there are gaps and the “edges” are a bit wide in places.

**FIGURE 7.3-4**

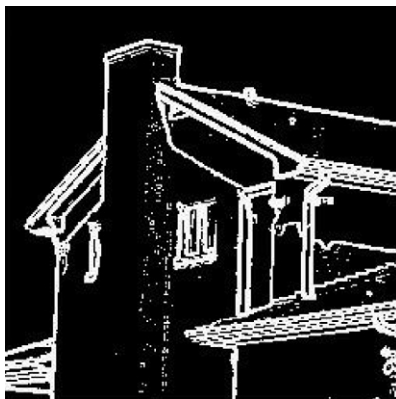
Thresholded output of vertical Sobel filter with $\gamma = 200$.

**FIGURE 7.3-5**

Output of threshold on the sum of absolute values of horizontal and vertical Sobel filters with $\gamma = 200$.

We can reduce the gaps by lowering the threshold; for example, at threshold $\gamma = 100$, we get the image in Figure 7.3-6 where the gaps on diagonal edges are much reduced. However, if there is some noise in the image, this improvement comes at a cost. In the image of Figure 7.3-7, we show the result of the Sobel edge detection with threshold $\gamma = 100$, when the original image contains a small amount of random noise, in this case Gaussian with $\mu = 0$ and $\sigma = 5$.

If we use a simple 3×3 box filter to smooth the noisy image prior to taking the Sobel filter analysis, we obtain the image in Figure 7.3-8 at the same threshold value of 100. More on edge detection is contained in *The Essential Guide to Image Processing* [8, Chapter 19].

**FIGURE 7.3-6**

Combined Sobel output with threshold $\gamma = 100$ for a noise-free image.

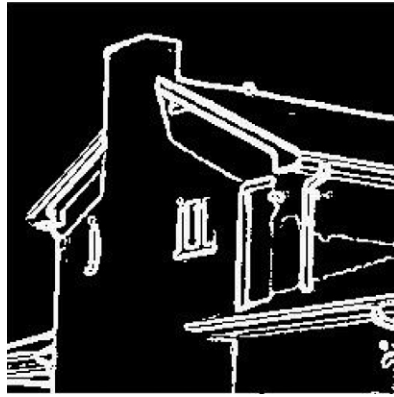
**FIGURE 7.3-7**

Threshold $\gamma = 100$ on combined Sobel output with a noisy image.

Edge Linking

The next step in a typical image analysis would be to attempt to link these edge pixels together to form curves or *lines*. One way to do this is called *edge relaxation* [9] in which the strengths of the candidate edge pixels (e.g., absolute values of gradient outputs) are locally used to increase or decrease *edge confidence* in an iterative way. The procedure is a simple yet effective way to fill in gaps in edges, and thus to form lines.

A formal edge-linking approach called *sequential edge linking* (SEL) was proposed by Eichel et al. [10] and then expanded to a multiresolution method in Cook and Delp [11]. This technique starts at a known edge pixel on the unthresholded gradient image and then hypothesizes potential edge paths using a log-likelihood method

**FIGURE 7.3-8**

Result of Sobel edge detection on the smoothed noisy image with threshold $\gamma = 100$.

and the Zigangirov-Jelinek (Z-J) search algorithm³ borrowed from channel-coding research, sometimes called the *stack algorithm*. The hard decision of thresholding is thus postponed until the path gets relatively long, thus giving increased robustness against image noise. On a rectangular lattice, there are eight nearest neighbors of each edge point, but a limited angular constraint ($< \pi/2$) on the developing paths reduces the number of pixels to search to three for the next edge on the path, or 3^n for an n -length edge curve, where the pointwise decisions are L(left), S(straight), or R(right). To avoid computational explosion, only the most promising paths, as measured by the likelihood-based *path metric*, are extended.

In an example from [11], the test image shown in Fig. 7.3-9a is immersed in independent and identically distributed (i.i.d.) Gaussian noise with mean zero, and standard deviation $\sigma = 40$. The magnitude of the gradient image is shown in Figure 7.3-9b. Figure 7.3-10 shows the SEL outputs: (a) from the basic algorithm presented in [10] and (b) from the multiresolution extension presented in [11]. Both these images are seen to show extremely good edge linking on this given test image.

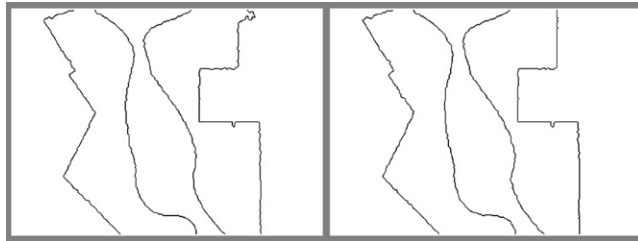
Segmentation

Regions or objects in images are often delineated by gray-level and color edges, but as we have seen, computed edge maps can often contain gaps and are not therefore closed. So they do not really provide a segmentation that constitutes the next level upward in image analysis (i.e., an image understanding). Thus various methods have been developed to segment the images directly using similarity of gray-level (and color) values and local textures. The simplest methods are based on the image

³This is a type of sequential decoding method. See Section 6.4 in Wozencraft and Jacobs [14]. Paths are searched in a depth-first manner using a discard criterion based on a log-likelihood ratio. Upon discard of the present hypothesized path, backtracking occurs. Computation and memory requirements are variable and generally grow with the noise level.

**FIGURE 7.3-9**

(a) The original 334×432 gray-level image and (b) the noisy gradient image (from Cook and Delp [11] © 1995 IEEE).

**FIGURE 7.3-10**

(a) SEL output from a two-level pyramid and (b) three-level multiresolution SEL output (from Cook and Delp [11] © 1995 IEEE).

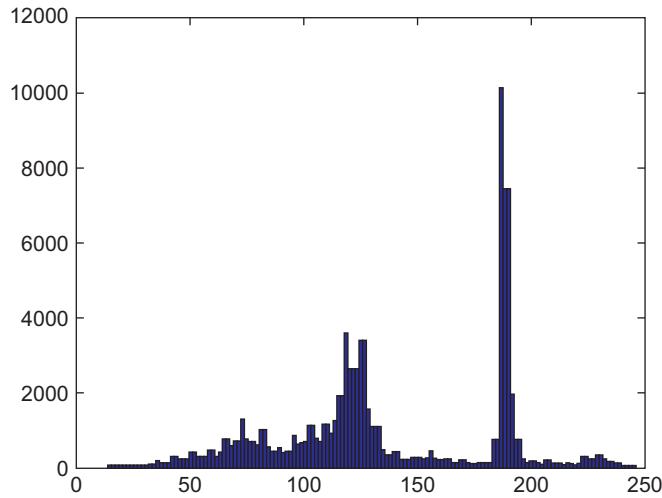
histogram. An automatic way to perform this is using the K-means algorithm. After regions are obtained, higher level semantic information can be used to compose these regions into meaningful objects.

Simple Thresholding

If we compute the image histogram, we may see that there are various minima apparent. These minima can be used to set thresholds for region segmentation. Manually setting thresholds will work well if the various objects in the image have sufficiently different gray levels and the gray-level variation within each object is sufficiently small.

Example 7.3-1: Manual Histogram Thresholding

Figure 7.3-11 shows the histogram of the House image from Figure 7.3-1. We can see two minima in this histogram located at gray levels 90 and 170, approximately. If we threshold the image at these points, we get the segmented image shown in Figure 7.3-12. Because we have used two thresholds on the histogram to do the separation, we have three segmented regions. The sky has been segmented together with the side edge of

**FIGURE 7.3–11**

Histogram of the gray-level House image.

**FIGURE 7.3–12**

Result of manually thresholding the gray-level house image at the two most prominent minima of the histogram, 90 and 170.

the roof⁴ and parts of the windows of the house. The house has been segmented into two regions, one for the sun lit part and the other for the shadow. A MATLAB routine, `HouseManThresh.m`, has been provided for easily experimenting with different thresholds and their number. ■

⁴While the side edge of the roof looks brighter than the sky in this segmented image, they are in fact the same gray value, 200. This is then an example of how surrounding affects perceived brightness.

K-means Algorithm

The K-means algorithm is a powerful method for image segmentation, which attempts to segment the pixel values of an image into K classes in such a way that the mean value of each class represents the image in a best sense as determined by a chosen distance measure, usually MSE.

Let the pixel values be given as $x(n_1, n_2)$ over the square region $[0, N_1 - 1] \times [0, N_2 - 1]$, and let there be K classes. We then want to find K numbers r_1, r_2, \dots, r_{K-1} that cluster the range of $x \in [0, 255]$ for standard 8-bit images. These numbers are the means of each cluster region. So we also need a set of K decision values d_k that define cluster regions $C_k = \{d_{k-1} \leq x(n_1, n_2) < d_k\}$, $k = 1$ to K , where $d_0 = 0$ and $d_K = 255$. Ideally we would like to choose the r_k such that the least-squares error

$$\sum_{k=1}^K \sum_{(n_1, n_2) \in C_k} [x(n_1, n_2) - r_k]^2 \quad (7.3-1)$$

is minimized. Effectively, we are replacing the actual pixel values in each cluster with representative numbers r_k in such a way that the least-squares error in the representation is minimized. This is a highly nonlinear problem to solve exactly, but we can see the two following properties of the optimal solution, called the *optimality conditions*.

Optimal Values for the C_k

Given a set of cluster centers (means), the minimal value of (7.3-1) will occur when values of generic pixels x are clustered to the nearest r_k with corresponding cluster region $C_k = \{d_{k-1} \leq x < d_k\}$. This will occur when we choose d_k to be halfway between the r_k values,

$$d_k = \frac{1}{2}(r_{k-1} + r_k), \quad (7.3-2)$$

because otherwise pixel x is closer to r_{k-1} and therefore should not be clustered to r_k .

Optimal Values for the r_k

Given a set of cluster regions C_k , the minimal value of the sum $\sum_{(n_1, n_2) \in C_k} [x(n_1, n_2) - r_k]^2$ will be obtained by the sample mean

$$r_k = \frac{1}{N_k} \sum_{(n_1, n_2) \in C_k} x(n_1, n_2), \quad (7.3-3)$$

with $N_k = |C_k|$ —the size of (number of pixels in) C_k .

Using these two optimality conditions, we can construct the following *K-means algorithm*, guaranteed to converge to at least a local minimum of (7.3-1).

1. Start with the initial guess of r_k (possibly obtained by inspection of the image histogram).
2. Calculate values for the C_k by using (7.3-2).
3. Calculate improved values for the r_k by applying (7.3-3).
4. Return to step 2 as long as a stopping criteria are not satisfied.

We can see that at each step the value of the metric (7.3-1) will not increase (most likely it will strictly decrease) so that a local minimum will be approached. We can

set the stopping criteria at a maximum number of iterations, or based on the decrease in (7.3–1) from the last iteration with a set minimal threshold. We show an example next using MATLAB.

Example 7.3–2: K-means

Using the MATLAB routine `kmeans`, which is part of the Statistical Toolbox, we segmented the 256×256 *cameraman* image into four classes. The original cameraman image is shown in Figure 7.3–13.



FIGURE 7.3–13

Original 256×256 cameraman image.

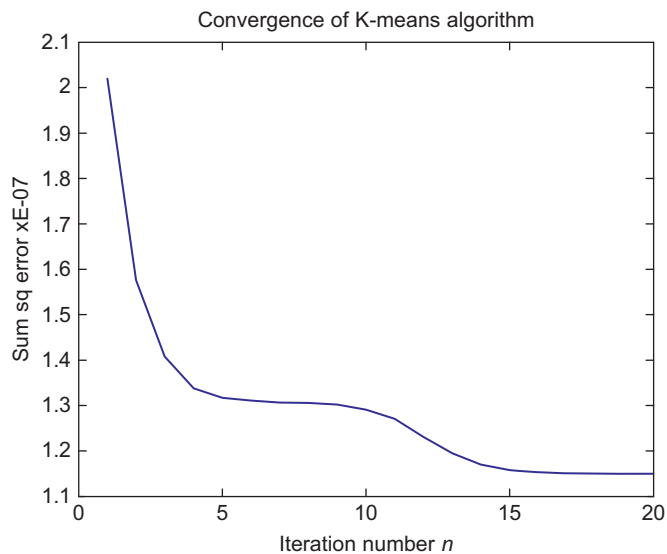
The initial values for the four mean values were randomly selected and the final converged values were 183.4, 113.9, 156.8, and 19.7. Note that these intensity values are not ordered because the initial values were chosen randomly. It took 20 iterations to converge, as shown in the plot of Figure 7.3–14.

The corresponding label image is shown in Figure 7.3–15. We can see that the sky and camera operator are segmented quite well. The grass in the foreground is less so, with spotty errors and gaps present. The actual representative values, or class means, are displayed as an image in Figure 7.3–16.

This clustering method can easily be extended to the case of vector \mathbf{x} where the cluster centers are necessarily also vectors \mathbf{r}_k and an appropriate distance measure $d(\mathbf{x}, \mathbf{r})$ must be defined. In this case, optimality condition 1 gets replaced by

$$C_k = \{\mathbf{x} | d(\mathbf{x}, \mathbf{r}_k) \leq d(\mathbf{x}, \mathbf{r}_m) \text{ for all } m \neq k\},$$

i.e. choose the nearest cluster C_k in distance d .

**FIGURE 7.3-14**

Plot of convergence of sum squares of K-means algorithm.

**FIGURE 7.3-15**

Image showing the four class indices resulting from the K-means algorithm.

The following example shows the K-means algorithm applied to a color image. Instead of scalar gray-level values, the algorithm now works on pixel vectors. The cluster centers are then distinct colors.

**FIGURE 7.3–16**

Image showing the class means—i.e., the representative values in each class.

Example 7.3–3: K-means for Color Images

We can also use the K-means algorithm for color images. Using the MATLAB routine `kmeans` again, we segment the 256×256 color house image into four classes, with random initialization points. It takes 20 iterations to converge to the four local mean RGB vectors:

$$c(1) = (156.3778, 108.1687, 102.5225),$$

$$c(2) = (157.7117, 194.5358, 217.0254),$$

$$c(3) = (214.1900, 216.6208, 217.0121),$$

$$c(4) = (95.7656, 62.5379, 78.5369).$$

The resulting index image is shown in [Figure 7.3–17](#). The resulting output color image is shown in [Figure 7.3–18](#). The distance used is sum of squares of differences in CIELAB space (cf. **CIELAB** in Section 6.5), called simply Lab space in MATLAB.

We could go on to apply the K-means method to small blocks of image values, say, 2×2 or 2×2 , thus beginning to explore segmentation based on local texture as well gray (color) pointwise values. Instead, we turn to introduce a method based on spatial location that results in the property of the segmented region(s) being connected.

Region Growing

The segmentation techniques discussed so far do not give attention to spatial location and so will tend to produce unconnected regions. Here, we look at a method

**FIGURE 7.3-17**

The index image of the K-means result for color image house.

**FIGURE 7.3-18**

Image showing the class means—i.e., the representative values in each class.

that builds in connectedness, called *region growing*. The following *region-growing algorithm* is based on the centroid linkage algorithm [12]:

1. Specify a comparison threshold T . Specify a seed pixel and note its value.
This is the first point in the region. Its value then becomes the region average.
2. Compare the nearest eight neighbor pixel values to the average value in the region.
If the difference is less than threshold T , then add such pixels to the region.
Otherwise, do not add the pixel.

3. Compare all nearest-neighbor pixels on the boundary of the region.
If the difference is less than the threshold, then add such pixels to the region.
Otherwise, do not add the pixel.
4. Go to step 3 if more pixels remain; otherwise stop.

Example 7.3–4: Growing a Region

We will apply a simple region-growing algorithm to the 512×480 gray-level *flower* image in Figure 7.3–19. The pixel values have been scaled to the interval $[0,1]$, and the point $(328, 341)$ located on a flower petal has been selected as the seed. We show two results, the first with threshold 0.08 in Figure 7.3–20 and the second with threshold 0.2 in Figure 7.3–21. We see that the threshold of 0.2 has done a pretty good job of segmenting



FIGURE 7.3–19

Gray-level 512×480 flower image.



FIGURE 7.3–20

Region grown from seed location $(328, 341)$ with threshold 0.08.

**FIGURE 7.3–21**

Region grown from seed location (328, 341) with threshold 0.2.

out the flower. (Performed using MATLAB Central routine `regiongrowing.m` provided by Dirk-Jan Kroon.)

Several seeds can be used in region growing to segment multiple objects or parts of one object. Significant interaction may be required to get the desired results. As with clustering using K-means, pixel values can be replaced by textures on small blocks to get a segmentation based on local texture in addition to pixel values. Similarly, RGB color images can be treated using a distance in the relevant color space. If segmenting for human viewing, then the more perceptually uniform color space CIELAB may be used.

7.4 OBJECT DETECTION

We now briefly look at two techniques for finding objects in images. First, we look at segmentation of a given image into a small number of regions based on a distance threshold. Then we look at a matched filtering approach to detecting objects, called *template matching*.

Object Segmentation

We can extend the region-growing algorithm to detect objects in color images. The basic algorithm must be modified to compute distance on the YUV or Lab coordinates. In addition, a raster scan of the pixels can detect *undefined* pixels in an outer loop added around the preceding algorithm. Each pass through this outer loop will define a separate region, and the threshold T will determine the number of regions

found. Additionally, some small regions are formed due to noise in the image. These must be merged into their nearest neighbors, in a *YUV* distance-defined region.

Example 7.4–1: Object Segmentation⁵

In this example, an object detection strategy was applied to a frame from the *Miss America* color video test clip of size 360×288 luma or *Y* pixels, with the *U* and *V* pixels subsampled by 2×2 . An appropriate Euclidean distance was formed based on the *YUV* pixel values. The threshold was set to $T = 40$ for this 8-bit image. Figure 7.4–1 shows the original frame.

Figure 7.4–2 shows a map of the four segments detected using this approach. We see that the face and part of the neck is detected as one object, with the



FIGURE 7.4–1


A frame from the Miss America test clip.



FIGURE 7.4–2

A four-region segmentation of a color image frame from the Miss America test clip.

⁵Example taken from the PhD thesis of Soo Chul Han [13].

shadow under the chin as a second object. The red tunic is detected as two separate objects, and the background is merged with her hair and black shirt. We thus see two types of errors: oversegmentation (red tunic) and overmerging (hair, background, black shirt). 

Some semantic knowledge can be used at this point to further improve segmentation accuracy if one is interested in true object recognition. In Chapter 11 on video processing, we will return to this example, using it as a basis for motion-based multiframe segmentation.

Template Matching

Often we have an example patch or small object given by an FIR template $p(n_1, n_2)$ that we seek to find or locate in an image x . A generally good way to accomplish this is with template matching. One way to accomplish this is with 2-D matched filtering, where we set the filter equal to the patch reflected through the origin,

$$h(n_1, n_2) = p(-n_1, -n_2),$$

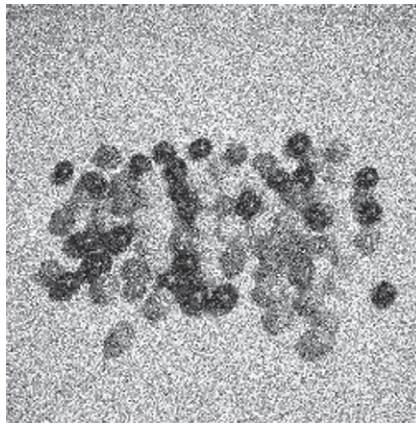
then the matched filter output is just $y = (h * x)(n_1, n_2)$. We look for peaks in this output and compare them to a fixed threshold γ ,

$$y(n_1, n_2) \geq \gamma.$$

If $y(n_1, n_2) \geq \gamma$, we say we have detected an instance of the object at location (n_1, n_2) in the image x ; otherwise the object is not detected. Under the assumption of a known object possibly (or not) present at a *known* location, and an image background composed of white Gaussian noise, the matched filter detector is known to be optimal [14]. While such ideal conditions are not often the case in image processing, still template matching is a commonly used and generally useful technique for locating known objects in cluttered backgrounds. We have already encountered a simple case of template matching in the edge detectors presented earlier, where the template was the edge model.

Example 7.4–2: Template Matching

In this example we seek to find the black jelly beans in a noisy image using template matching. The noisy image is shown in Figure 7.4–3 and was produced by adding computer-generated white Gaussian noise ($\sigma = 50$) to the 256×256 monochrome image shown in Figure 7.4–4. The template image for this example was formed by cropping out the bottom rightmost black jelly bean to give a 18×15 - pixel template shown in Figure 7.4–5. After normalizing this template to have zero mean, it was reflected through the origin and convolved with the noisy image in the MATLAB program

**FIGURE 7.4-3**

Noisy jelly beans image.

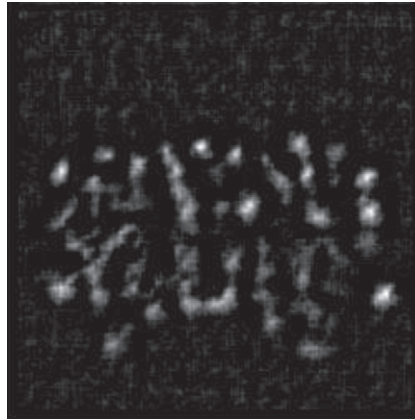
**FIGURE 7.4-4**

Original jelly beans image.

**FIGURE 7.4-5**

Black jelly bean template.

TemplateMatch.m, available at this textbook's Web site. The filter output image is shown in Figure 7.4-6. When subjected to a threshold of 125, this image produced the logical mask image shown in Figure 7.4-7. Figure 7.4-8 shows an overlay of the

**FIGURE 7.4-6**

Matched filter output image.

**FIGURE 7.4-7**

Threshold output image, where white (250) indicates a black jelly bean and black (0) indicates its absence.

detector-thresholded output over the original image. We see a generally successful detection of the black jelly beans, although with some false alarms, mostly due to the large noise level. The threshold can be adjusted to trade off the number of black beans detected versus the false alarms (i.e., white or gray miss-detected). You can try out this routine yourself with a lower noise level and different threshold.

**FIGURE 7.4–8**

Detector output superposed over the original jelly bean image.

CONCLUSIONS

In this chapter, we introduced some simple image processing filters that are commonly used for enhancement and noted their shortcomings from the frequency-response viewpoint. Design techniques from Chapter 5 can usefully be applied here to improve on some of these simple image processing filters. We used some of the simple filters to perform basic image denoising. We looked at the median filter, a simple nonlinear filter, and applied it to denoising problems. We then looked at introductory image analysis, first finding edges and then linking them together. We also looked at segmentation of regions and objects as a second level in image analysis. We will find several of these techniques applied in later chapters on image and video processing. Finally, we introduced the problem of object detection and template matching. Basic image processing techniques are covered by Bovik in Chapter 2.1 of [15].

PROBLEMS

1. The box filter was introduced in [Section 7.1](#) as a simple FIR filter that finds wide use in image processing practice. Here, we consider a square $L \times L$ box filter and implement it recursively to reduce the required number of additions.
 - (a) Consider an odd-length 1-D $L = 2M + 1$ point box filter with *unscaled* output

$$Ly(n) = \sum_{k=n-M}^{n+M} x(k),$$

and show that this sum can be realized recursively by

$$Ly(n) = Ly(n-1) + x(n+M) - x(n-M-1).$$

How many adds and multiplies per point are required for this 1-D filter?

- (b) Find a 2-D method to realize the square $L \times L$ box filter for odd L . How much *intermediate storage* is required by your method? (Intermediate storage is the temporary storage needed for the processing. It does not include any storage that may be needed to store either the input or output arrays.)
2. For the horizontal derivative 3×3 FIR approximation, called the Sobel operator, consider calculating samples of its frequency response with a 2-D DFT program. Assume the DFT is of size $N \times N$ for some large value of N .
 - (a) Where can you place the Sobel operator coefficients in the square $[0, N-1]^2$ if you are only interested in samples of the magnitude response?
 - (b) Where must you place the Sobel operator coefficients in the square $[0, N-1]^2$ if you are interested in samples of the amplitude of the Fourier transform?
3. Image processing texts recommend that the Sobel operator (filter)

$$\begin{Bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{Bmatrix}$$

should only be used on a smoothed image. So, consider the simple 3×3 smoothing filter

$$\begin{Bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$

with output taken at the center point. Let us smooth first and then apply the Sobel operator to the smoothed output.

- (a) What is the size of the resulting combined FIR filter?
- (b) What is the combined filter's impulse response?
- (c) Find the frequency response of the combined filter.
- (d) For a spatial domain or direct implementation, is it more efficient to apply these two 3×3 filters in series or to apply the combined filter? Why?
4. Use the available MATLAB routine `HouseSobEdgeNoise` and the `WinDesC` filter design routine to determine a "best" smoothing lowpass filter in terms of cutoff frequency for a 5×5 FIR filter. Measure "best" by the visual quality of the resulting edge mask in terms of connectedness, thinness, gaps or missed edge pixels, and false edge pixels. Of course, your findings will be very subjective. First, vary cutoff frequency for the fixed threshold value 100. Then experiment locally about

this “optimum” for various thresholds in the range 80–150. Compare the result to that of the 3×3 box filter.

5. Prove optimality condition 2 (7.3–3) for the K-means algorithm by finding the optimal value of r_k to minimize the sum

$$\sum_{(n_1, n_2) \in C_k} [x(n_1, n_2) - r_k]^2$$

for a given C_k .

6. Use the MATLAB routine `kmeans.m` (from the Statistical Toolbox) to find a locally optimal segmentation of the monochrome *Barbara* image based on 2×2 blocks into four and eight clusters. You may use the provided routine `kmeans4HouseRandom.m` as a base.
7. Experiment with other color spaces for cluster-based segmentation using the color 512×512 Lena image and the MATLAB `kmeans.m` routine. The image processing toolbox has color conversion routines for sRGB, XYZ, and CIELAB (LAB).

REFERENCES

- [1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd Ed., Prentice-Hall, Upper Saddle River, NJ, 2002.
- [2] P. J. Burt and E. H. Adelson, “The Laplacian Pyramid as a Compact Image Code,” *IEEE Trans. Comm.*, vol. COM-31, pp. 532–540, April 1983.
- [3] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
- [4] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [5] H. Stark and J. W. Woods, *Probability and Random Processes with Appl. to Signal Process.*, 3rd Ed., Prentice-Hall, Upper Saddle River, NJ, 2002.
- [6] J. W. Goodman, *Intro. to Fourier Optics*, 3rd Ed., Roberts and Company Publishers, Greenwood Village, CO, 2005.
- [7] Y.-S. Lee, “Graphical Demonstration of an Optimality Property of the Median,” *The American Statistician*, vol. 49, no. 4, pp. 369–372, 1995.
- [8] A. C. Bovik, Ed., *The Essential Guide to Image Processing*, 2nd Ed., Elsevier Academic Press, Burlington, MA, 2009.
- [9] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [10] P. H. Eichel, E. J. Delp, K. Koral, and A. J. Buda, “A Method for a Fully Automatic Definition of Coronary Arterial Edges from Cineangiograms,” *IEEE Trans. Medical Imaging*, vol. 7, pp. 313–320, December 1988.
- [11] G. W. Cook and E. J. Delp, “Multiresolution Sequential Edge Linking,” *IEEE Int. Conf. on Image Process (ICIP)*, Washington DC, October 1995.

- [12] R. Haralick and L. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Reading, MA, 1992.
- [13] S.-C. Han, *Object-Based Representation and Compression of Image Sequences*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1997.
- [14] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, John Wiley & Sons, New York, 1965.
- [15] A. C. Bovik, Ed., *Handbook of Image and Video Process.*, 2nd. Ed., Elsevier Academic Press, Burlington, MA, 2005.

Image Estimation and Restoration

8

Here, we apply the linear systems and basic image processing knowledge of the previous chapters to modeling, developing solutions, and experimental testing of algorithms for two dominant problems in digital image processing: *image estimation* and *image restoration*. By image estimation we mean the case where a clean image has been contaminated with noise, usually through sensing, transmission, or storage. We will treat the independent and additive noise case. The second problem, image restoration, means that in addition to the noise, there is some blurring due to motion or lack of focus. We attempt to “restore” the image in this case. Of course, the restoration will only be approximate.

We first develop the theory of linear estimation in two dimensions and then present some example applications for monochrome image processing problems. We then look at some space-variant and nonlinear estimators. This is followed by a section on image and/or blur model parameter identification (estimation) and combined image restoration. We next look at some more recent work, including non-Bayesian approaches. We then present a third problem, *image superresolution*, which extends image restoration to the case where several similar copies of an image are available, and a spatial upsampling is required. Finally, we make some brief comments on extensions of the methods presented in this chapter to color images.

All of the image data in this chapter are thought to be “density data” (i.e., image data after gamma compensation). This is the common space used in image processing, probably for two reasons: first, this space (domain) is more subjectively uniform than the original intensity space, and second, most often the exact nature of the gamma compensation is unknown.

8.1 TWO-DIMENSIONAL RANDOM FIELDS¹

A random sequence in two dimensions is called a *random field*. Images that display clearly random characteristics include sky with clouds, textures of various types, and sensor noise. It may perhaps be surprising that we also model general images that

¹This section requires a background in 1-D random sequences and processes. An introduction and summary is provided in the appendix to this chapter.

are unknown as random fields with an eye to their estimation, restoration, and, in a later chapter, transmission and storage. But this is a necessary first step in designing systems that have an optimality across a class of images sharing common statistics. The theory of 2-D random fields builds upon random process and probability theory. Many references exist including [1]. We start with some basic definitions.

Definition 8.1–1: Random Field

A 2-D random field $x(n_1, n_2)$ is a mapping from a probability *sample space* Ω to the class of 2-D sequences. As such, for each outcome $\zeta \in \Omega$, we have a deterministic sequence, and for each location (n_1, n_2) , we have a random variable.

We are not using capital letters to distinguish between a random variable and the value it takes on, i.e., $X = x$. This is because we are reserving capitalization for matrices and Fourier transforms. For a random sequence $x(n_1, n_2)$, we define the following low-order moments:

Definition 8.1–2: Second-Order Moments

The *mean function* of a random sequence $x(n_1, n_2)$ is denoted

$$\mu_x(n_1, n_2) \triangleq E\{x(n_1, n_2)\}.$$

Correlation function,

$$R_x(n_1, n_2; m_1, m_2) \triangleq E\{x(n_1, n_2)x^*(m_1, m_2)\}.$$

Covariance function,

$$K_x(n_1, n_2; m_1, m_2) \triangleq E\{x_c(n_1, n_2)x_c^*(m_1, m_2)\},$$

where $x_c(n_1, n_2) \triangleq x(n_1, n_2) - \mu_x(n_1, n_2)$ and is called the *centered version* of x .

We immediately have the following relation between these first- and second-order moment functions,

$$K_x(n_1, n_2; m_1, m_2) = R_x(n_1, n_2; m_1, m_2) - \mu_x(n_1, n_2)\mu_x^*(m_1, m_2),$$

and we also define the *variance function*

$$\begin{aligned}\sigma_x^2(n_1, n_2) &\triangleq K_x(n_1, n_2; n_1, n_2) \\ &= \text{var}\{x(n_1, n_2)\} = E\{|x_c(n_1, n_2)|^2\},\end{aligned}$$

and note

$$\sigma_x^2(n_1, n_2) \geq 0.$$

When all the statistics of a random field do not change with position (n_1, n_2) , we say that the random field is *homogeneous*, analogously to the stationarity property from random sequences [1]. The homogeneity assumption is often relied upon in

order to estimate the needed statistical quantities from a given realization or sample sequence or image. Often the set of images used to estimate these statistics is called the *training set*.

Definition 8.1–3: Homogeneous Random Field

A random field is homogeneous when the N th-order joint probability density function (pdf) is invariant with respect to the N positions, and this holds for all positive integers N ; that is, for all N and locations \mathbf{n}_i , we have

$$f_x(x(\mathbf{n}_1), x(\mathbf{n}_2), \dots, x(\mathbf{n}_N)) = f_x(x(\mathbf{n}_1 + \mathbf{m}), x(\mathbf{n}_2 + \mathbf{m}), \dots, x(\mathbf{n}_N + \mathbf{m})),$$

independent of the shift vector \mathbf{m} .²

Usually we do not have such a complete description of a random field as is afforded by a complete set of joint pdf's of all orders, and so must resort to partial descriptions, often in terms of the low-order moment functions. This situation calls for the following classification that is much weaker than (strict) homogeneity.

Definition 8.1–4: Wide-Sense Homogeneous

A 2-D random field is wide-sense homogeneous (wide-sense stationary) if

1. $\mu_x(n_1, n_2) = \mu_x(0, 0) = \text{constant}$, and
2. $R_x(n_1 + m_1, n_2 + m_2; n_1, n_2) = R_x(m_1, m_2; 0, 0)$, independent of n_1 and n_2 .

In the strict-sense homogeneous case, for notational simplicity we define the constant mean $\mu_x \triangleq \mu_x(0, 0)$ and the two-parameter correlation function $R_x(m_1, m_2) \triangleq R_x(m_1, m_2; 0, 0)$ and covariance function $K_x(m_1, m_2) \triangleq R_x(m_1, m_2; 0, 0) - \mu_x^2$.

Example 8.1–1: Independent and Identically Distributed Gaussian Noise

Let $w(n_1, n_2)$ be independent and identically distributed (i.i.d.) Gaussian noise with mean function $\mu_w(n_1, n_2) = \mu_w$, and standard deviation $\sigma_w > 0$. Clearly, we have wide-sense homogeneity here. The two-parameter correlation function is given as

$$R_w(m_1, m_2) = \sigma_w^2 \delta(m_1, m_2) + \mu_w^2$$

and covariance function

$$K_w(m_1, m_2) = \sigma_w^2 \delta(m_1, m_2).$$

²Be careful. While the notation $f_x(x(\mathbf{n}_1), x(\mathbf{n}_2), \dots, x(\mathbf{n}_N))$ seems friendly enough, it is really short-hand for $f_x(x(\mathbf{n}_1), x(\mathbf{n}_2), \dots, x(\mathbf{n}_N); \mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_N)$, where the locations are given explicitly. This extended notation *must* be used when we evaluate f_x at specific choices for the $x(\mathbf{n}_i)$.

The first-order pdf becomes

$$f_x(x(\mathbf{n})) = \frac{1}{\sqrt{2\pi}\sigma_w} \exp - \frac{(x(\mathbf{n}) - \mu_w)^2}{2\sigma_w^2} \\ \sim N(\mu_w, \sigma_w^2).$$

The i.i.d. noise field is called *white noise* when its mean is zero. ■

Filtering a 2-D Random Field

Let $G(\omega_1, \omega_2)$ be the frequency response of a spatial filter, with impulse response $g(n_1, n_2)$. Consider the convolution of the filter impulse response with an i.i.d. noise field $w(n_1, n_2)$ whose mean is μ_w . Calling the output random field $x(n_1, n_2)$, we have

$$x(n_1, n_2) = g(n_1, n_2) * w(n_1, n_2) \\ = \sum_{k_1, k_2} g(k_1, k_2) w(n_1 - k_1, n_2 - k_2).$$

We can then compute the mean and covariance function of x as follows:

$$\begin{aligned} \mu_x(n_1, n_2) &= E\{x(n_1, n_2)\} \\ &= E \left\{ \sum_{k_1, k_2} g(k_1, k_2) w(n_1 - k_1, n_2 - k_2) \right\} \\ &= \sum_{k_1, k_2} g(k_1, k_2) E\{w(n_1 - k_1, n_2 - k_2)\} \\ &= \sum_{k_1, k_2} g(k_1, k_2) \mu_w(n_1 - k_1, n_2 - k_2) \\ &= \sum_{k_1, k_2} g(k_1, k_2) \mu_w \\ &= G(0, 0) \mu_w. \end{aligned}$$

The covariance function of the generated random field $x(n_1, n_2)$ is calculated as

$$\begin{aligned} K_x(n_1, n_2; m_1, m_2) &= E\{x_c(n_1, n_2) x_c^*(m_1, m_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} g(k_1, k_2) g^*(l_1, l_2) E\{w_c(n_1 - k_1, n_2 - k_2) \\ &\quad \times w_c^*(m_1 - l_1, m_2 - l_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} g(k_1, k_2) g^*(l_1, l_2) K_w(n_1 - k_1, n_2 - k_2; m_1 - l_1, m_2 - l_2) \\ &= g(n_1, n_2) * K_w(n_1, n_2; m_1, m_2) * g^*(m_1, m_2) \\ &= g(n_1, n_2) * \sigma_w^2 \delta(n_1 - m_1, n_2 - m_2) * g^*(m_1, m_2). \end{aligned} \quad (8.1-1)$$

Such a “coloring” of the i.i.d. noise is often useful in generating models for real correlated noise. A warning on our notation in (8.1–1): the first $*$ denotes 2-D convolution on the (n_1, n_2) variables, while the second $*$ denotes 2-D convolution on the (m_1, m_2) variables.

Generalizing a bit, we can write the output of a filter $H(\omega_1, \omega_2)$ with general colored noise input $x(n_1, n_2)$ as

$$\begin{aligned} y(n_1, n_2) &= h(n_1, n_2) * x(n_1, n_2) \\ &= \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2). \end{aligned}$$

This could be a filtering of the random field model we just created. Calculating the mean function of the output y , we find

$$\begin{aligned} \mu_y(n_1, n_2) &= E\{y(n_1, n_2)\} \\ &= E\left\{ \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) \right\} \\ &= \sum_{k_1, k_2} h(k_1, k_2) E\{x(n_1 - k_1, n_2 - k_2)\} \\ &= \sum_{k_1, k_2} h(k_1, k_2) \mu_x(n_1 - k_1, n_2 - k_2) \\ &= h(n_1, n_2) * \mu_x(n_1, n_2). \end{aligned}$$

And for the correlation function,

$$\begin{aligned} R_y(n_1, n_2; m_1, m_2) &= E\{y(n_1, n_2) y^*(m_1, m_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) h^*(l_1, l_2) E\{x(n_1 - k_1, n_2 - k_2) \\ &\quad \times x^*(m_1 - l_1, m_2 - l_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) h^*(l_1, l_2) R_x(n_1 - k_1, n_2 - k_2; m_1 - l_1, m_2 - l_2) \\ &= h(n_1, n_2) * R_x(n_1, n_2; m_1, m_2) * h^*(m_1, m_2), \end{aligned}$$

and covariance function,

$$\begin{aligned} K_y(n_1, n_2; m_1, m_2) &= E\{y_c(n_1, n_2) y_c^*(m_1, m_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) h^*(l_1, l_2) E\{x_c(n_1 - k_1, n_2 - k_2) \\ &\quad \times x_c^*(m_1 - l_1, m_2 - l_2)\} \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) h^*(l_1, l_2) K_x(n_1 - k_1, n_2 - k_2; m_1 - l_1, m_2 - l_2) \\ &= h(n_1, n_2) * K_x(n_1, n_2; m_1, m_2) * h^*(m_1, m_2). \end{aligned}$$

If we specialize to the homogeneous case, we obtain

$$\mu_y(n_1, n_2) = \sum_{k_1, k_2} h(k_1, k_2) \mu_x$$

or, more simply,

$$\mu_y = H(0, 0) \mu_x.$$

The correlation function is given as

$$\begin{aligned} R_y(m_1, m_2) &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) h^*(l_1, l_2) R_x(m_1 - k_1 + l_1, m_2 - k_2 + l_2) \\ &= \sum_{k_1, k_2} \sum_{l_1, l_2} h(k_1, k_2) R_x(m_1 - k_1 + l_1, m_2 - k_2 + l_2) h^*(l_1, l_2) \\ &= h(m_1, m_2) * R_x(m_1, m_2) * h^*(-m_1, -m_2) \\ &= (h(m_1, m_2) * h^*(-m_1, -m_2)) * R_x(m_1, m_2). \end{aligned} \quad (8.1-2)$$

Similarly, for the covariance function,

$$K_y(m_1, m_2) = (h(m_1, m_2) * h^*(-m_1, -m_2)) * K_x(m_1, m_2).$$

Taking Fourier transforms, we can move to the *power spectral density* (PSD) domain

$$S_x(\omega_1, \omega_2) \triangleq \sum_{m_1, m_2} R_x(m_1, m_2) \exp -j(m_1 \omega_1 + m_2 \omega_2)$$

and obtain the Fourier transform of (8.1-2) as

$$\begin{aligned} S_y(\omega_1, \omega_2) &= H(\omega_1, \omega_2) S_x(\omega_1, \omega_2) H^*(\omega_1, \omega_2) \\ &= |H(\omega_1, \omega_2)|^2 S_x(\omega_1, \omega_2). \end{aligned} \quad (8.1-3)$$

Example 8.1-2: Power Spectrum of Images

One equation is often used to represent the PSD of a typical image [2],

$$S_x(\omega_1, \omega_2) = \frac{K}{(1 + (\omega_1^2 + \omega_2^2)/\omega_0^2)^{3/2}} \quad \text{for } |\omega_1|, |\omega_2| \leq \pi,$$

and was used in [3] with $K = \pi/42.19$ to model a video conferencing image frame. The resulting log plot from MATLAB is given in Figure 8.1-1 (ref. Spect Image .m). A 64×64 -point discrete Fourier transform (DFT) was used, and zero frequency is in the middle of the plot, at $k_1 = k_2 = 32$. Note that the image appears to be very lowpass in nature, even on this decibel or log scale. Note that this PSD is not rational and so does not correspond

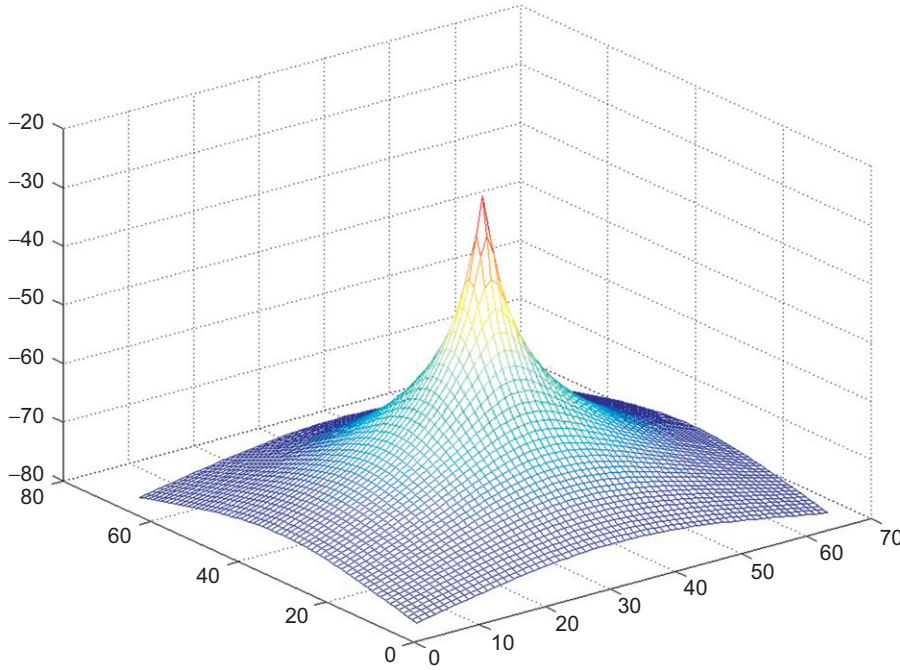


FIGURE 8.1–1

Log or dB plot of example spectra.

to a finite 2-D difference equation model, although it could be well approximated by such. ■

Autoregressive Random Signal Models

A particularly useful model for random signals and noises is the autoregressive (AR) model. Mathematically this is a 2-D difference equation driven by white noise. As such it generates a correlated zero-mean field, which can be colored by the choice of its coefficients. The 2-D AR model is given as

$$x(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2} x(n_1 - k_1, n_2 - k_2) + w(n_1, n_2).$$

We can find the coefficients a_{k_1, k_2} by solving the 2-D linear prediction problem

$$\hat{x}(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2} x(n_1 - k_1, n_2 - k_2), \quad (8.1-4)$$

which can be done using the *orthogonality principle* of estimation theory [1].

Theorem 8.1–1: Optimal Linear Prediction

The (1,0)-step linear prediction coefficients in (8.1–4) that minimize the mean-square prediction error,

$$E\{|x(n_1, n_2) - \hat{x}(n_1, n_2)|^2\},$$

can be determined by the orthogonality principle, which states that the error $e \triangleq \hat{x} - x$ must be *orthogonal* to the data used in the linear prediction:

$$e(n_1, n_2) \perp x(n_1 - k_1, n_2 - k_2) \text{ for } (k_1, k_2) \in \mathcal{R}_a - (0, 0).$$

For two random variables, orthogonal means that their correlation is zero, so we have $E\{e(n_1, n_2)x^*(n_1 - k_1, n_2 - k_2)\} = 0$ for $(k_1, k_2) \in \mathcal{R}_a - (0, 0)$. To actually find the coefficients, we start with

$$E\{e(n_1, n_2)x^*(n_1 - k_1, n_2 - k_2)\} = 0,$$

and substitute $e = \hat{x} - x$, to obtain

$$E\{\hat{x}(n_1, n_2)x^*(n_1 - k_1, n_2 - k_2)\} = E\{x(n_1, n_2)x^*(n_1 - k_1, n_2 - k_2)\},$$

or

$$\begin{aligned} E\left\{\sum_{l_1, l_2} a_{l_1, l_2} x(n_1 - l_1, n_2 - l_2)x^*(n_1 - k_1, n_2 - k_2)\right\} &= E\{x(n_1, n_2)x^*(n_1 - k_1, n_2 - k_2)\}, \\ \sum_{l_1, l_2} a_{l_1, l_2} R_x(n_1 - l_1, n_2 - l_2; n_1 - k_1, n_2 - k_2) &= R_x(n_1, n_2; n_1 - k_1, n_2 - k_2) \\ &\text{for } (k_1, k_2) \in \mathcal{R}_a - (0, 0), \end{aligned}$$

which is an ordinary set of \mathcal{N} linear equations in \mathcal{N} unknown coefficients—i.e., the number of points in the set $\mathcal{R}_a - (0, 0)$. This assumes we know the correlation function R_x for the indicated range. Once these coefficients are found, the mean-square power (variance) of $e(n_1, n_2)$ can be determined. ■

In the homogeneous case, these equations simplify to

$$\sum_{l_1, l_2} a_{l_1, l_2} R_x(k_1 - l_1, k_2 - l_2) = R_x(k_1, k_2) \text{ for } (k_1, k_2) \in \mathcal{R}_a - (0, 0).$$

Defining a coefficient vector \mathbf{a} of dimension \mathcal{N} , these equations can easily be put into matrix form. The resulting correlation matrix \mathbf{R} will be invertible if the correlation function R_x is positive definite. The mean-square prediction error is then easily

obtained as

$$\begin{aligned}
 E\{|e(n_1, n_2)|^2\} &= E\{e(n_1, n_2)(\hat{x} - x)^*(n_1, n_2)\} \\
 &= -E\{e(n_1, n_2)x^*(n_1, n_2)\} \\
 &= -E\{(\hat{x} - x)(n_1, n_2)x^*(n_1, n_2)\} \\
 &= E\{|x|^2(n_1, n_2)\} - E\{\hat{x}(n_1, n_2)x(n_1, n_2)^*\} \\
 &= \sigma_x^2(n_1, n_2) - \sum_{l_1, l_2} a_{l_1, l_2} R_x(n_1 - l_1, n_2 - l_2; n_1, n_2),
 \end{aligned}$$

which specialized to the homogeneous case becomes

$$\sigma_e^2 = \sigma_x^2 - \sum_{l_1, l_2} a_{l_1, l_2} R_x(-l_1, -l_2).$$

Example 8.1–3: (1 × 1)-order QP Predictor

Let the predictor coefficient support of the 2-D linear predictor be given as

$$\mathcal{R}_a - (0, 0) = \{(1, 0), (0, 1), (1, 1)\}.$$

Assume the random field x is homogeneous with the zero mean and correlation (covariance) function given as $R_x(m_1, m_2)$. Then the three so-called Normal equations are given in matrix form as

$$\begin{bmatrix} R_x(0, 0) & R_x(1, -1) & R_x(0, -1) \\ R_x(-1, 1) & R_x(0, 0) & R_x(-1, 0) \\ R_x(0, 1) & R_x(1, 0) & R_x(0, 0) \end{bmatrix} \begin{bmatrix} a_{1,0} \\ a_{0,1} \\ a_{1,1} \end{bmatrix} = \begin{bmatrix} R_x(1, 0) \\ R_x(0, 1) \\ R_x(1, 1) \end{bmatrix}.$$

8.2 ESTIMATION FOR RANDOM FIELDS

Here, we consider the problem of estimating a random field x from observations of another random field y . We assume that x and y are zero mean and write the estimate as

$$\hat{x}(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_h} h_{k_1, k_2} y(n_1 - k_1, n_2 - k_2).^3$$

Using the orthogonality principle, with $e = \hat{x} - x$, we have

$$\hat{x}(n_1, n_2) - x(n_1, n_2) \perp \{y(n_1 - k_1, n_2 - k_2) \text{ for } (k_1, k_2) \in \mathcal{R}_h\},$$

³If x and y had nonzero means, μ_x and μ_y , respectively, then the appropriate estimate would have the form $\hat{x} = \mu_x + h * (y - \mu_y)$.

which becomes

$$E\{\hat{x}(n_1, n_2)y^*(n_1 - k_1, n_2 - k_2)\} = E\{x(n_1, n_2)y^*(n_1 - k_1, n_2 - k_2)\}, \text{ or}$$

$$E\left\{\sum_{l_1, l_2} h_{l_1, l_2} y(n_1 - l_1, n_2 - l_2) y^*(n_1 - k_1, n_2 - k_2)\right\} = E\{x(n_1, n_2)y^*(n_1 - k_1, n_2 - k_2)\},$$

$$\sum_{l_1, l_2} h_{l_1, l_2} R_{yy}(n_1 - l_1, n_2 - l_2; n_1 - k_1, n_2 - k_2) = R_{xy}(n_1, n_2; n_1 - k_1, n_2 - k_2)$$

for $(k_1, k_2) \in \mathcal{R}_h$.

Specialized to the homogeneous case, we have

$$\sum_{(l_1, l_2) \in \mathcal{R}_h} h_{l_1, l_2} R_{yy}(k_1 - l_1, k_2 - l_2) = R_{xy}(k_1, k_2) \text{ for } (k_1, k_2) \in \mathcal{R}_h. \quad (8.2-1)$$

Infinite Observation Domain

In the case where the observation region \mathcal{R}_h is infinite, (8.2-1) becomes a convolution. For example, if $\mathcal{R}_h = (-\infty, +\infty)^2$, we get

$$h(n_1, n_2) * R_{yy}(n_1, n_2) = R_{xy}(n_1, n_2), \text{ where } -\infty < n_1, n_2 < +\infty.$$

We can express this convolution in the frequency domain,

$$H(\omega_1, \omega_2) S_{yy}(\omega_1, \omega_2) = S_{xy}(\omega_1, \omega_2),$$

or

$$H(\omega_1, \omega_2) = \frac{S_{xy}(\omega_1, \omega_2)}{S_{yy}(\omega_1, \omega_2)},$$

at those frequencies where $S_{yy} > 0$,⁴ which is the general equation for the 2-D noncausal (unrealizable) Wiener filter. We consider a few special cases.

1. $y = x + n$ with $x \perp n$ (i.e., x and n are orthogonal),

$$S_{xy}(\omega_1, \omega_2) = S_{xx}(\omega_1, \omega_2) \text{ and}$$

$$S_{yy}(\omega_1, \omega_2) = S_{xx}(\omega_1, \omega_2) + S_{nn}(\omega_1, \omega_2),$$

so that the Wiener filter is

$$H(\omega_1, \omega_2) = \frac{S_{xx}(\omega_1, \omega_2)}{S_{xx}(\omega_1, \omega_2) + S_{nn}(\omega_1, \omega_2)},$$

the so-called *estimation* case. Notice that the filter is approximately 1 at those frequencies where the signal-to-noise ratio (SNR) is high, with the SNR defined as $S_{xx}(\omega_1, \omega_2)/S_{nn}(\omega_1, \omega_2)$. The filter then attenuates other values.

⁴At those frequencies where $S_{yy} = 0$, the exact value of H does not matter.

2. $y = g * x + n$ with $x \perp n$, then

$$S_{xy}(\omega_1, \omega_2) = S_{xx}(\omega_1, \omega_2)G^*(\omega_1, \omega_2)$$

and

$$S_{yy}(\omega_1, \omega_2) = |G(\omega_1, \omega_2)|^2 S_{xx}(\omega_1, \omega_2) + S_{nn}(\omega_1, \omega_2),$$

so that the Wiener filter is

$$H(\omega_1, \omega_2) = \frac{G^*(\omega_1, \omega_2)S_{xx}(\omega_1, \omega_2)}{|G(\omega_1, \omega_2)|^2 S_{xx}(\omega_1, \omega_2) + S_{nn}(\omega_1, \omega_2)},$$

the so-called *restoration* case. Notice that the filter looks like an inverse filter at those frequencies where the SNR is high. The filter tapers off and provides less gain at other values.

3. $y = u + n$ with $u \perp n$, and we want to estimate $x \triangleq b * u$. In this case the Wiener filter is just b convolved with the estimate in case 1,

$$H(\omega_1, \omega_2) = \frac{B(\omega_1, \omega_2)S_{uu}(\omega_1, \omega_2)}{S_{uu}(\omega_1, \omega_2) + S_{nn}(\omega_1, \omega_2)}.$$

4. $y(n_1, n_2) = x(n_1 - 1, n_2)$, the linear prediction case, as treated earlier.

A good application of case 3 would be the estimation of a derivative of a signal.

Wiener Filter—Alternative Derivation

A direct derivation of the Wiener filter through the concept of a 2-D *whitening filter* is possible. We first give a brief summary of 2-D spectral factorization, whose factor will yield the needed whitening filter. Then we re-derive the noncausal and go on to find the causal Wiener filter.

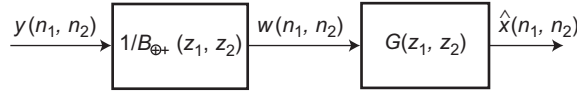
Theorem 8.2–1: Spectral Factorization

Given a homogeneous random field $x(n_1, n_2)$ with PSD $S_{xx}(\omega_1, \omega_2) > 0$ on $[-\pi, +\pi]^2$, there exists a *spectral factorization*

$$S_{xx}(\omega_1, \omega_2) = \sigma^2 B_{\oplus+}(\omega_1, \omega_2) B_{\ominus-}(\omega_1, \omega_2),$$

with $\sigma > 0$, and $B_{\oplus+}(z_1, z_2)$ is stable and causal with a stable causal inverse. ■

In terms of Z-transforms, we have $B_{\ominus-}(z_1, z_2) = B_{\oplus+}(z_1^{-1}, z_2^{-1})$, where we assume real coefficients in the spectral factors. Unfortunately, even when the PSD $S_{xx}(\omega_1, \omega_2)$ is rational, the spectral factors $B_{\oplus+}$ and $B_{\ominus-}$ will generally be infinite order which can then be used as ideal functions in a 2-D filter design approximation. Alternatively, the factors can be related to the linear prediction filters of the previous section, and approximation can be obtained in that way too. More on 2-D spectral factorization is contained in [4, 5].

**FIGURE 8.2-1**

Whitening filter realization of Wiener filter.

Consider observations consisting of the homogenous random field $y(n_1, n_2)$, and assume the causal spectral factor of $S_{yy}(\omega_1, \omega_2)$ is given as $B_{\oplus+}(z_1, z_2)$. Since this factor has a stable and causal inverse, we can use $B_{\oplus+}^{-1}(z_1, z_2)$ to whiten the spectra of y and obtain whitened output $w(n_1, n_2)$, with variance σ_w^2 , often called the 2-D *innovations sequence*. We can then just as well base our estimate on w and filter it with $G(z_1, z_2)$ to obtain the estimate $\hat{x}(n_1, n_2)$, as shown in Figure 8.2-1.

We can define the estimation error at the output of G as

$$e(n_1, n_2) \triangleq x(n_1, n_2) - \sum_{l_1, l_2} g(l_1, l_2) w(n_1 - l_1, n_2 - l_2), \quad (8.2-2)$$

and then express the mean-square error (MSE) of the estimate in the real-valued case as

$$\begin{aligned} E\{e^2(n_1, n_2)\} &= E \left\{ \left[x(n_1, n_2) - \sum_{l_1, l_2} g(l_1, l_2) w(n_1 - l_1, n_2 - l_2) \right]^2 \right\} \\ &= R_{xx}(0, 0) - 2E \left\{ \sum_{l_1, l_2} g(l_1, l_2) x(n_1, n_2) w(n_1 - l_1, n_2 - l_2) \right\} \\ &\quad + E \left\{ \left[\sum_{l_1, l_2} g(l_1, l_2) w(n_1 - l_1, n_2 - l_2) \right]^2 \right\}. \end{aligned}$$

Next, we use the whiteness of the innovations $w(n_1, n_2)$ to obtain

$$\begin{aligned} E\{e^2(n_1, n_2)\} &= R_{xx}(0, 0) - 2 \sum_{l_1, l_2} g(l_1, l_2) R_{xw}(l_1, l_2) + \sigma_w^2 \sum_{l_1, l_2} g^2(l_1, l_2) \\ &= R_{xx}(0, 0) + \sum_{l_1, l_2} [\sigma_w g(l_1, l_2) - R_{xw}(l_1, l_2)/\sigma_w]^2 \\ &\quad - \frac{1}{\sigma_w^2} \sum_{l_1, l_2} R_{xw}^2(l_1, l_2), \end{aligned} \quad (8.2-3)$$

where the last step comes from completing the square. At this point we observe that minimizing over choice of g is simple because it only affects the middle term, which

clearly has its minimum at 0, which is easily obtained by setting

$$g(l_1, l_2) = \frac{1}{\sigma_w^2} R_{xw}(l_1, l_2).$$

The overall linear minimum MSE filter is obtained by convolving the whitening filter with g . In the Z-transform domain we have

$$\begin{aligned} G(z_1, z_2) &= \frac{1}{\sigma_w^2} S_{xw}(z_1, z_2) \\ &= \frac{1}{\sigma_w^2} \frac{S_{xy}(z_1, z_2)}{B_{\oplus+}(z_1^{-1}, z_2^{-1})}. \end{aligned}$$

So

$$\begin{aligned} H(z_1, z_2) &= \frac{1}{B_{\oplus+}(z_1, z_2)} G(z_1, z_2) \\ &= \frac{S_{xy}(z_1, z_2)}{\sigma_w^2 B_{\oplus+}(z_1, z_2) B_{\oplus+}(z_1^{-1}, z_2^{-1})} \\ &= \frac{S_{xy}(z_1, z_2)}{S_{yy}(z_1, z_2)}. \end{aligned}$$

NSHP Causal Wiener Filter

If we restrict the sum over l_1 and l_2 in (8.2–2) to the infinite NSHP region

$$\mathcal{R}_{\oplus+} = \{n_1 \geq 0, n_2 \geq 0\} \cup \{n_1 < 0, n_2 > 0\},$$

we have a causal Wiener filter. Equation (8.2–3) changes to

$$\begin{aligned} E\{e^2(n_1, n_2)\} &= R_{xx}(0, 0) + \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} [\sigma_w g(l_1, l_2) - R_{xw}(l_1, l_2)/\sigma_w]^2 \\ &\quad - \frac{1}{\sigma_w^2} \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} R_{xw}^2(l_1, l_2), \end{aligned}$$

which is minimized at

$$g(l_1, l_2) = \begin{cases} \frac{1}{\sigma_w^2} R_{xw}(l_1, l_2), & (l_1, l_2) \in \mathcal{R}_{\oplus+}, \\ 0, & \text{else.} \end{cases}$$

In the Z-transform domain, we have

$$G(z_1, z_2) = \frac{1}{\sigma_w^2} [S_{xw}(z_1, z_2)]_{\oplus+},$$

where the subscript notation $\oplus+$ on the cross-power spectra S_{xw} means that we only include the part that has support on $\mathcal{R}_{\oplus+}$. We can write the full expression for the

2-D causal Wiener filter as

$$\begin{aligned}
 G(z_1, z_2) &= \frac{1}{\sigma_w^2} S_{xw}(z_1, z_2) \\
 &= \frac{1}{\sigma_w^2 B_{\oplus+}(z_1, z_2)} [S_{xw}(z_1, z_2)]_{\oplus+} \\
 &= \frac{1}{\sigma_w^2 B_{\oplus+}(z_1, z_2)} \left[\frac{S_{xy}(z_1, z_2)}{B_{\oplus+}(z_1^{-1}, z_2^{-1})} \right]_{\oplus+}.
 \end{aligned}$$

Again, this is an infinite-order ideal filter. For approximation, any of the filter design methods we have considered can be used. However, if the design is in the frequency domain, then both magnitude and phase error must be taken into account. An example was shown at the end of Chapter 5 in the design of a fully recursive Wiener filter. Incidentally, the Wiener filter has been found particularly effective for joint electro-optical or lens-DSP system design [6].

8.3 TWO-DIMENSIONAL RECURSIVE ESTIMATION

First, we review recursive estimation in one dimension, where we focus on the Kalman filter as the linear estimation solution for an AR signal model. After a brief review of the 1-D Kalman filter, we then extend this approach to two dimensions.

1-D Kalman Filter

We start with an AR signal model in the 1-D case,

$$x(n) = \sum_{k=1}^M c_k x(n-k) + w(n),$$

where $n \geq 0$. The mean $\mu_w = 0$ and correlation $R_w(m) = \sigma_w^2 \delta(m)$ for the white model input noise w . The initial condition is *initial rest*: $x(-1) = x(-2) = \dots = x(-M) = 0$. We do not observe the signal x directly. Instead, we see the signal x convolved with an M -tap FIR filter h in the presence of an observation noise v that is orthogonal to the signal generation noise w . The so-called *observation equation* is then given as

$$y(n) = h(n) * x(n) + v(n), \quad \text{for } n \geq 0.$$

Re-expressing this model in vector form, we have, upon defining the M -dimensional signal column vectors

$$\mathbf{x}(n) \triangleq [x(n), x(n-1), \dots, x(n-M+1)]^T, \quad \mathbf{w}(n) \triangleq [w(n), 0, \dots, 0]^T,$$

and system matrix

$$\mathbf{C} \triangleq \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_M \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{Q}_w \triangleq \begin{bmatrix} \sigma_w^2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix},$$

the signal state equation

$$\mathbf{x}(n) = \mathbf{C}\mathbf{x}(n-1) + \mathbf{w}(n), \text{ for } n \geq 0, \text{ subject to } \mathbf{x}(-1) = \mathbf{0}.$$

The advantage of this vector notation is that the M th-order case can be treated with vector algebra just as simply as the first-order case. The scalar observation equation, based on a linear combination of the elements of the state vector, can be written in vector form as

$$y(n) = \mathbf{h}^T \mathbf{x}(n) + v(n), \text{ for } n \geq 0,$$

by defining the column vector \mathbf{h} as

$$\mathbf{h} \triangleq [h_0, h_1, \dots, h_{M-1}]^T,$$

where $h_l = h(l)$, $l = 0, \dots, M-1$. In these equations the observation noise v is a white noise also $R_v(m) = \sigma_v^2 \delta(m)$, but orthogonal to white model noise w (i.e., $v(n) \perp x(m)$ for all n and m).

Since we have only specified the second order properties (i.e., mean and correlation functions) of these signals and noises, we do not have a complete stochastic characterization of this estimation problem. However, we can find the best estimator that is constrained to be linear from this information, as we did with the Wiener filter nonrecursive formulation in the last section. The result is the so-called *linear minimum mean-square error* (LMMSE) estimator.

Some comments are in order:

1. The signal model is strictly Markov- p with $p = M$ only when the signal generating noise w is Gaussian distributed.
2. When we have also that the observation noise is Gaussian, it turns out that the LMMSE estimator is globally optimal in the unconstrained *minimum mean-square error* (MMSE) case also. This occurs because, in the joint Gaussian case, the optimal MMSE solution is linear in the data.
3. In applications, the support of the smoothing function h may not match the order of the state equation. In that case, either equation may be padded with zeros to come up with a match. Equivalently, we set $M = \max[\text{these two integers}]$.
4. In many applications, the matrix \mathbf{C} is time-variant. Dynamic physics-based models are used in the mechanical and aerospace industries to stochastically model vehicles and aircraft. The 1-D Kalman filter is widely used there.

From the study of estimation theory [1], we know that the MMSE solution to the problem is equivalent to finding the conditional mean

$$E\{\mathbf{x}(n)|y(n), y(n-1), \dots, y(0)\} \triangleq \hat{\mathbf{x}}_a(n),$$

called the *filtering estimate*. As a by-product, we also obtain the so-called *one-step predictor estimate*

$$E\{\mathbf{x}(n)|y(n-1), y(n-2), \dots, y(0)\} \triangleq \hat{\mathbf{x}}_b(n).$$

The solution for this estimation problem, when the signal and noise are jointly Gaussian distributed, is contained in many texts on 1-D statistical signal processing, (e.g. [1]).

1-D Kalman Filter Equations

Prediction: $\hat{\mathbf{x}}_b(n) = \mathbf{C}\hat{\mathbf{x}}_a(n-1), \quad n \geq 0.$

Updates: $\hat{\mathbf{x}}_a(n) = \hat{\mathbf{x}}_b(n) + \mathbf{g}(n)[y(n) - \mathbf{h}^T \hat{\mathbf{x}}_b(n)].$

Here, $\mathbf{g}(n)$ is the *Kalman gain* vector that scales the observation prediction error term $y(n) - \mathbf{h}^T \hat{\mathbf{x}}_b(n)$ prior to employing it as an additive update term to the state prediction vector $\hat{\mathbf{x}}_b(n)$. This gain vector $\mathbf{g}(n)$ is determined from *error covariance equations* of the state vector $\mathbf{x}(n)$, which proceed via the nonlinear iterations:

Error covariance: $\mathbf{P}_b(n) = \mathbf{C}\mathbf{P}_a(n-1)\mathbf{C}^T + \mathbf{Q}_w, \quad n > 0,$
 $\mathbf{P}_a(n) = (\mathbf{I} - \mathbf{g}(n)\mathbf{h}^T)\mathbf{P}_b(n), \quad n \geq 0.$

Here, $\mathbf{P}_b(n)$ and $\mathbf{P}_a(n)$ are the *error-covariance matrices*, before and after updating, respectively. Their definition is

$$\mathbf{P}_b(n) \triangleq E\{(\mathbf{x}(n) - \hat{\mathbf{x}}_b(n))(\mathbf{x}(n) - \hat{\mathbf{x}}_b(n))^T\} \text{ and}$$

$$\mathbf{P}_a(n) \triangleq E\{(\mathbf{x}(n) - \hat{\mathbf{x}}_a(n))(\mathbf{x}(n) - \hat{\mathbf{x}}_a(n))^T\}.$$

The gain vector then is given in terms of these as

Gain vector: $\mathbf{g}(n) \triangleq \mathbf{P}_b(n)\mathbf{h} \left(\mathbf{h}^T \mathbf{P}_b(n)\mathbf{h} + \sigma_v^2 \right)^{-1}, \quad n \geq 0.$

We note from the AR signal model that $x(0) = w(0)$, or equivalently $\mathbf{x}(0) = (w(0), 0, \dots, 0)^T$, from which follows the initial conditions $\hat{\mathbf{x}}_b(0) = \mathbf{0}$ and $\mathbf{P}_b(n) = \mathbf{Q}_w$ for the preceding filter equations. For a complete derivation, in the special no-blur case (i.e., $\mathbf{h} = \delta$), the reader is referred to Section 9.2 in [1], with the general blur case treated in problem 9.10 of that text.

Since the underlying signal model and observation model are scalar, these vector equations can be recast into equivalent scalar equations to offer more insight, which we do next.

Scalar Kalman Filtering Equations

Predictor:

$$\hat{x}_b^{(n)}(n) = \sum_{l=1}^M c_l \hat{x}_a^{(n-1)}(n-l), \quad \text{with } \hat{x}_b^{(n)}(m) = \hat{x}_a^{(n-1)}(m) \text{ for } m < n. \quad (8.3-1)$$

Updates:

$$\hat{x}_a^{(n)}(m) = \hat{x}_b^{(n)}(m) + g^{(n)}(n-m)[y(n) - \sum_{l=0}^{M-1} h_l \hat{x}_b^{(n)}(n-l)], \quad \text{for } n - (M-1) \leq m \leq n. \quad (8.3-2)$$

We see that there is first a prediction estimate $\hat{x}_b^{(n)}(n)$ based on the *past* observations $y(m)$, $m < n$, but after this there are multiple updated estimates $\hat{x}_a^{(n)}(m)$ based on the current observation $y(n)$, beginning at time $n = m$ and proceeding to future time $n = m + (M-1)$. The estimate resulting from the first update is called the *Kalman filter* estimate, and the remaining ones are called *Kalman smoother* estimates. The former is a causal estimate, while the latter involve a fixed lag behind the observations $y(n)$. Note that the estimates must not get worse, and normally will get better, as they are successively updated. This follows from the corresponding optimality of this recursive estimation procedure for the MSE criteria. Thus we expect the fixed lag smoother estimate of maximum lag $\hat{x}_a^{(n)}(n - (M-1))$ to be the best one.

The scalar error-covariance equations become the following:

$$\text{Before updates: } R_b^{(n)}(n; k) = \sum_{l=1}^M c_l R_a^{(n)}(n-l; m), \quad \text{for } k < n,$$

$$R_b^{(n)}(n; n) = \sum_{l=1}^M c_l R_a^{(n)}(n; n-l) + \sigma_w^2.$$

$$\text{After updates: } R_a^{(n)}(k; l) = R_b^{(n)}(k; l) - g^{(n)}(n-k)$$

$$\cdot \sum_{o=0}^{M-1} h(o) R_b^{(n)}(n-o; l), \quad \text{for } n - (M-1) \leq k, l \leq n.$$

The Kalman gain vector becomes in scalar form the *gain array*,

$$g^{(n)}(k) = \sum_{l=0}^{M-1} h(l) R_b^{(n)}(n-l; n-k) \left/ \left(\sum_{l=0}^{M-1} \sum_{o=0}^{M-1} h(l) h(o) R_b^{(n)}(n-l; n-o) + \sigma_v^2 \right) \right.,$$

$$\text{for } 0 \leq k \leq M-1.$$

2-D Kalman Filtering

If we consider a 2-D or spatial AR signal model

$$x(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_{\oplus+}} c_{k_1, k_2} x(n_1 - k_1, n_2 - k_2) + w(n_1, n_2), \quad (8.3-3)$$

and scalar observation model

$$y(n_1, n_2) = \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2) + v(n_1, n_2), \quad (8.3-4)$$

over a finite observation region \mathcal{O} , we can process these observations in row-scanning order (also called a raster scan) to map them to an equivalent 1-D problem. The 2-D AR model then maps to a 1-D AR model, but with a much larger state vector with large internal gaps. If the $\oplus+$ AR model is of order (M, M) , and the observation region is a square of side N (i.e., $\mathcal{O} = [0, N - 1]^2$), then the equivalent 1-D AR model is of order $O(MN)$. This situation is illustrated in Figure 8.3-1. Looking at the scalar filtering equations (8.3-1) and (8.3-2), we see that the prediction term is still of low order $O(M^2)$, but the update term is of order $O(MN)$. The Kalman gain array is then also of order $O(MN)$, so that the nonlinear error-covariance equations must be run to calculate each of these update gain coefficients and are hence also of order $O(MN)$. Since an image size is normally very large compared to the AR signal model orders, this situation is completely unsatisfactory. We thus conclude that the very efficient computational advantage of the Kalman filter in 1-D is a special case and is lost in going to higher dimensions. Still, there are useful approximations that can be employed based on the observation that, for most reasonable image signal models, the gain terms should be mainly confined to a small region surrounding the current observations. This process then results in the reduced update Kalman filter that has found good use in image processing when a suitable image signal model (8.3-3) and image blur model (8.3-4) are available.

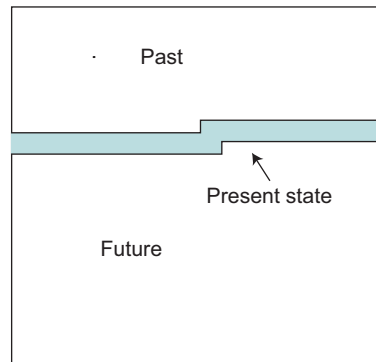


FIGURE 8.3-1

Illustration of the global state vector of a spatial Kalman filter.

The scalar filtering equations for the 2-D Kalman filter are as follows:

Prediction:

$$\begin{aligned}\hat{x}_b^{(n_1, n_2)}(n_1, n_2) &= \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} c_{l_1 l_2} \hat{x}_a^{(n_1-1, n_2)}(n_1 - l_1, n_2 - l_2), \\ \hat{x}_b^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_a^{(n_1-1, n_2)}(k_1, k_2), \quad \text{for } (k_1, k_2) \in \mathcal{S}_{\oplus+}(n_1, n_2).\end{aligned}$$

Updates:

$$\begin{aligned}\hat{x}_a^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_b^{(n_1, n_2)}(k_1, k_2) + g^{(n_1, n_2)}(n_1 - k_1, n_2 - k_2) \\ &\quad \left[y(n_1, n_2) - \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} h(l_1, l_2) \hat{x}_b^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2) \right], \\ &\quad \text{for } (k_1, k_2) \in \mathcal{S}_{\oplus+}(n_1, n_2),\end{aligned}$$

where $\mathcal{S}_{\oplus+}(n_1, n_2)$ is the 2-D *global state region* that must be updated at scan location (n_1, n_2) as shown in the gray area in Figure 8.3–1. The corresponding error-covariance equations are as follows:

Before updates:

$$\begin{aligned}R_b^{(n_1, n_2)}(n_1, n_2; k_1, k_2) &= \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} c_{l_1 l_2} R_a^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2; k_1, k_2), \\ &\quad \text{for } (k_1, k_2) \in \mathcal{S}_{\oplus+}(n_1, n_2), \\ R_b^{(n_1, n_2)}(n_1, n_2; n_1, n_2) &= \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} c_{l_1 l_2} R_a^{(n_1, n_2)}(n_1, n_2; n_1 - l_1, n_2 - l_2) + \sigma_w^2.\end{aligned}$$

After updates:

$$\begin{aligned}R_a^{(n_1, n_2)}(k_1, k_2; l_1, l_2) &= R_b^{(n_1, n_2)}(k_1, k_2; l_1, l_2) - g^{(n_1, n_2)}(n_1 - k_1, n_2 - k_2) \\ &\quad \cdot \sum_{(o_1, o_2) \in \mathcal{S}_h} h(o_1, o_2) R_b^{(n_1, n_2)}(n_1 - o_1, n_2 - o_2; l_1, l_2), \\ &\quad \text{for all } (k_1, k_2) \text{ and } (l_1, l_2) \in \mathcal{S}_{\oplus+}(n_1, n_2).\end{aligned}$$

The 2-D gain array is given as

$$\begin{aligned}g^{(n_1, n_2)}(k_1, k_2) &= \frac{\sum_{(l_1, l_2) \in \mathcal{S}_h} h(l_1, l_2) R_b^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2; n_1 - k_1, n_2 - k_2)}{\sum_{(l_1, l_2) \in \mathcal{S}_h} \sum_{(o_1, o_2) \in \mathcal{S}_h} h(o_1, o_2) h(l_1, l_2) R_b^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2; n_1 - o_1, n_2 - o_2) + \sigma_v^2} \\ &\quad \text{for all } (k_1, k_2) \in \mathcal{S}_{\oplus\oplus}(n_1, n_2).\end{aligned}\tag{8.3–5}$$

In the case of no blurring (i.e., $h = \delta$), the 2-D Kalman gain array simplifies to

$$g^{(n_1, n_2)}(k_1, k_2) = R_b^{(n_1, n_2)}(n_1, n_2; n_1 - k_1, n_2 - k_2) / \left(R_b^{(n_1, n_2)}(n_1, n_2; n_1, n_2) + \sigma_v^2 \right).$$

Reduced Update Kalman Filter

As a simple approximation to the preceding spatial Kalman filtering equations, we see that the problem is with the update, as the prediction is already very efficient. So, we decide to update only nearby previously processed data points in our raster scan. We will definitely update those points needed directly in the upcoming predictions but will omit update of points further away. Effectively, we define a local update region $\mathcal{U}_{\oplus+}(n_1, n_2)$ of order $O(M^2)$ with the property that

$$\mathcal{R}_{\oplus+}(n_1, n_2) \subset \mathcal{U}_{\oplus+}(n_1, n_2) \subset \mathcal{S}_{\oplus+}(n_1, n_2).$$

This is an approximation since all the points in $\mathcal{S}_{\oplus+}(n_1, n_2)$ will be used in some future prediction. However, if the current update is only significant for points in the $O(M^2)$ neighborhood of (n_1, n_2) , then omitting the updates of points further away should have negligible effect. The choice of how large to make the update region $\mathcal{U}_{\oplus+}(n_1, n_2)$ has been a matter of experimental determination.

The spatial recursive filtering equations for the *reduced update Kalman filter* (RUKF) then become the manageable set:

Prediction:

$$\begin{aligned}\hat{x}_b^{(n_1, n_2)}(n_1, n_2) &= \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} c_{l_1 l_2} \hat{x}_a^{(n_1-1, n_2)}(n_1 - l_1, n_2 - l_2), \\ \hat{x}_b^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_a^{(n_1-1, n_2)}(k_1, k_2), \quad \text{for } (k_1, k_2) \in \mathcal{U}_{\oplus+}(n_1, n_2).\end{aligned}$$

Updates:

$$\begin{aligned}\hat{x}_a^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_b^{(n_1, n_2)}(k_1, k_2) + g^{(n_1, n_2)}(n_1 - k_1, n_2 - k_2) \\ &\quad \left[y(n_1, n_2) - \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} h(l_1, l_2) \hat{x}_b^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2) \right], \\ &\quad \text{for } (k_1, k_2) \in \mathcal{U}_{\oplus+}(n_1, n_2),\end{aligned}$$

with computation per data point of $O(M^2)$. In this way, experience has shown that a good approximation to spatial Kalman filtering can often be obtained [7].

Approximate RUKF

While the RUKF results in a very efficient set of prediction and update equations, the error covariance equations are still very high order computationally. This is because the error covariance of each updated estimate must be updated with each nonupdated estimate—i.e., for $(k_1, k_2) \in \mathcal{U}_{\oplus+}(n_1, n_2)$ and, most importantly, all $(l_1, l_2) \in \mathcal{S}_{\oplus+}(n_1, n_2)$. To further address this problem, we approximate

the RUKF by also omitting these error-covariance updates beyond a larger *covariance update region* $\mathcal{T}_{\oplus+}(n_1, n_2)$ that is still $O(M^2)$. Experimentally, it is observed that the resulting *approximate RUKF* can be both very computationally efficient as well as very close to the optimal linear MMSE estimator [7]. However, the choice of the appropriate size for the update region $\mathcal{U}_{\oplus+}$ and the covariance update region $\mathcal{T}_{\oplus+}$ is problem dependent, and so some trial and error may be necessary to get near the best result. These various regions then satisfy the inclusion relations

$$\mathcal{R}_{\oplus+}(n_1, n_2) \subset \mathcal{U}_{\oplus+}(n_1, n_2) \subset \mathcal{T}_{\oplus+}(n_1, n_2) \subset \mathcal{S}_{\oplus+}(n_1, n_2),$$

where only the last region $\mathcal{S}_{\oplus+}$, the global state region, is $O(N^2)$.

Steady-State RUKF

For the preceding approximations to work, we generally need a stable AR signal model. In this case, experience has shown that the approximate reduced update filter converges to an LSI filter as $(n_1, n_2) \nearrow (\infty, \infty)$. For typical AR image models, as few as 5–10 rows and columns are often sufficient. In that case, the gain array becomes independent of the observation position, and to an excellent approximation, the error-covariance equations no longer have to be calculated. The resulting LSI filter, for either RUKF or approximate RUKF, then consists of two steps:

Prediction:

$$\begin{aligned} \hat{x}_b^{(n_1, n_2)}(n_1, n_2) &= \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} c_{l_1 l_2} \hat{x}_a^{(n_1-1, n_2)}(n_1 - l_1, n_2 - l_2), \\ \hat{x}_b^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_a^{(n_1-1, n_2)}(k_1, k_2), \quad \text{for } (k_1, k_2) \in \mathcal{U}_{\oplus+}(n_1, n_2). \end{aligned}$$

Updates:

$$\begin{aligned} \hat{x}_a^{(n_1, n_2)}(k_1, k_2) &= \hat{x}_b^{(n_1, n_2)}(k_1, k_2) + g(n_1 - k_1, n_2 - k_2) \\ &\quad \left[y(n_1, n_2) - \sum_{(l_1, l_2) \in \mathcal{R}_{\oplus+}} h(l_1, l_2) \hat{x}_b^{(n_1, n_2)}(n_1 - l_1, n_2 - l_2) \right], \\ &\quad \text{for } (k_1, k_2) \in \mathcal{U}_{\oplus+}(n_1, n_2). \end{aligned}$$

LSI Estimation and Restoration Examples with RUKF

We present three examples of the application of the steady-state RUKF linear recursive estimator to image estimation and restoration problems. The first example is an estimation problem.

Example 8.3–1: Image Estimation

Figure 8.3–2a shows an original image *Lena*, which is 256×256 and monochrome. In Figure 8.3–2b is seen the same image plus white Gaussian noise added in the contrast domain (also called density domain) of Chapter 6. Since the filters assume zero mean, the global mean is subtracted from the noisy image prior to processing and then added in after processing to produce the final estimate. Figure 8.3–3 shows the steady-state RUKF estimate based on the noisy data at $SNR = 10$ dB in Figure 8.3–2b. The SNR of the estimate is 14.9 dB, so that the *SNR improvement* (ISNR) is 4.9 dB. These results come from [8]. We can see the visible smoothing effect of this filter. ■

**FIGURE 8.3–2**

(a) 256×256 *Lena*—original. (b) *Lena* plus noise at 10-dB input SNR.

**FIGURE 8.3–3**

RUKF estimate of *Lena* from 10-dB noisy data.

The next two examples consider the case where the blur function h is operative (i.e., image restoration). The first one is for a linear 1-D blur, which can simulate linear camera or object motion. If a camera moves uniformly in the horizontal direction for exactly M pixels during its exposure time, then an $M \times 1$ FIR blur can result,

$$h(n) = \frac{1}{M} [u(n) - u(n - M)].$$

Nonuniform motion will result in other calculable but linear blur functions.

Example 8.3–2: Image Restoration from 1-D Blur

Figure 8.3–4 shows the 256×256 monochrome cameraman image blurred horizontally by a 10×1 FIR blur. The input blurred SNR (BSNR) = 40 dB. Again, the global mean is first subtracted from the noisy image prior to processing and added in after processing to produce the final estimate. Figure 8.3–5 shows the result of the 3-gain restoration algorithm making use of RUKF. The SNR improvement is 12.5 dB. Note that while there is considerable increase in sharpness, there is clearly some ringing evident. There is more on this example in [8]. This example uses the inhomogeneous image model of next Section 8.4.

The next example shows RUKF restoration performance for a simulated uniform area blur of size 7×7 . Such area blurs are often used to simulate the camera's lack of perfect focus, or simply being out of focus.⁵ Also, certain types of sensor



FIGURE 8.3–4

Cameraman blurred by horizontal FIR blur of length 10. BSNR = 40 dB.

⁵Fourier optics theory shows that lens blur can be approximated with LSI convolution in the original image *intensity space*. Since we are typically filtering in a (partially unknown) gamma-compensated space, LSI convolution is only a first-order or linear approximation, perhaps best suited only to low-contrast image data.

**FIGURE 8.3–5**

Inhomogeneous Gaussian using 3-gains and residual model.

vibration can cause the information obtained at a pixel to be close to a local average of incident light. In any case, a uniform blur is a challenging case, compared to more realistic tapered blurs that are more concentrated for the same total support.

Example 8.3–3: Image Restoration from Area Blur

This is an example of image restoration from a 7×7 uniform blur. The simulation was done in floating point, and white Gaussian noise was added to the blurred cameraman image to produce a $BSNR = 40$ dB. Again, the global mean is first subtracted from the noisy image prior to processing and added in after processing to produce the final estimate. We investigate the effect of boundary conditions on this restoration RUKF estimate. We use the image model of [9]. Figure 8.3–6 shows the restoration using a circulant blur, where we can see ringing coming from the picture boundaries where the circulant blur does not match the assumed linear blur of the RUKF, and Figure 8.3–7 shows the restoration using a linear blur model, where we notice much less picture-boundary induced ringing due to the assumed linear blur model of the RUKF. The update regions for both RUKF steady-state estimates were $(-12, 10, 12)$, meaning $(-west, +east, +north)$, a 12×12 update region “northwest” of, and including the present observation, plus 10 more columns “northeast.” For Figure 8.3–6, the ISNR was 2.6 dB, or leaving out a border of 25 pixels on all sides 3.8 dB. For Figure 8.3–7, the corresponding $ISNR = 4.4$ dB and leaving out the 25-pixel boundary 4.4 dB. In both cases, the covariance update region was four

**FIGURE 8.3-6**

RUKF restoration using circulant blur model from area blur.

**FIGURE 8.3-7**

RUKF restoration from linear area blur model.

columns wider than the update region. Interestingly, the corresponding improvement of the DFT-implemented Wiener filter [9] was reported at 3.9 dB. We should note that the RUKF results are not very good for small error covariance update regions. For example, in [9] they tried $(-6, 2, 6)$ for both the update and covariance update region and found only 0.6-dB improvement. But then again, 7×7 is a very large uniform blur support. ■

Reduced Order Model

An alternative to the RUKF is the *reduced order model* (ROM) method. The ROM approach to Kalman filtering is well established [10], with applications in image processing [11]. In that method, approximation is made to the signal (image) model to constrain its global state to order $O(M^2)$, and then there is no need to reduce the update. The ROM Kalman filter (ROMKF) of Angwin and Kaufman [12] has been prized for its ready adaptation capabilities. A relation between ROMKF and RUKF has been explained in [13].

8.4 INHOMOGENEOUS GAUSSIAN ESTIMATION

Here, we extend our AR image model to the inhomogeneous case by including a local mean and local variance [8]. Actually, these two must be estimated from the noisy data, but usually the noise remaining in them is rather small. So we write our *inhomogeneous Gaussian image model* as

$$x(n_1, n_2) \triangleq x_r(n_1, n_2) + \mu_x(n_1, n_2),$$

where $\mu_x(n_1, n_2)$ is the space-variant mean function of $x(n_1, n_2)$, and $x_r(n_1, n_2)$ is a *residual model* given as

$$x_r(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_c - (0,0)} c_{k_1, k_2} x_r(n_1 - k_1, n_2 - k_2) + w_r(n_1, n_2), \quad (8.4-1)$$

where, as usual, $w_r(n_1, n_2)$ is a zero-mean, white Gaussian noise with variance function $\sigma_{w_r}^2(n_1, n_2)$, a space-variant function. The *model noise* w_r describes the uncertainty in the residual image signal model.

Now, considering the estimation case, the observation equation is

$$y(n_1, n_2) = x(n_1, n_2) + v(n_1, n_2),$$

which upon subtraction of the mean $\mu_y(n_1, n_2)$ becomes

$$y_r(n_1, n_2) = x_r(n_1, n_2) + v(n_1, n_2), \quad (8.4-2)$$

where $y_r(n_1, n_2) \triangleq y(n_1, n_2) - \mu_y(n_1, n_2)$, using the fact that the observation noise is assumed zero mean. We then apply the RUKF to the model and observation pair, (8.4-1) and (8.4-2). The RUKF will give us an estimate of $x_r(n_1, n_2)$, and the residual RUKF estimate of the signal then becomes

$$\hat{x}(n_1, n_2) = \hat{x}_r(n_1, n_2) + \mu_x(n_1, n_2).$$

In a practical application, we would have to estimate the space-variant mean $\mu_x(n_1, n_2)$, and this can be accomplished in a simple way, using a box filter (cf. Section 7.1) as

$$\hat{\mu}_x(n_1, n_2) \triangleq \frac{1}{(2M+1)^2} \sum_{(m_1, m_2) \in [-M, +M]^2} y(n_1 - m_1, n_2 - m_2),$$

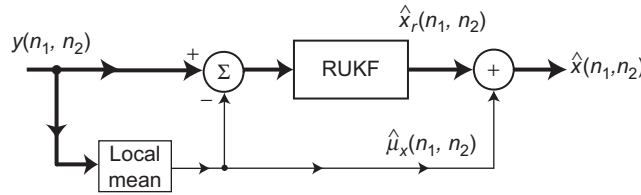


FIGURE 8.4–1

System diagram for inhomogeneous Gaussian estimation with RUKF.

and relying on the fact that $\mu_x = \mu_y$ since the observation noise v is zero mean. We can also estimate the local variance of the residual signal x_r through an estimate of the local variance of the residual observations y_r , as

$$\hat{\sigma}_{y_r}^2(n_1, n_2) \triangleq \frac{1}{(2M+1)^2} \sum_{(m_1, m_2) \in [-M, +M]^2} (y(n_1 - m_1, n_2 - m_2) - \hat{\mu}_x(n_1, n_2))^2$$

and

$$\hat{\sigma}_{x_r}^2(n_1, n_2) \triangleq \max\{\hat{\sigma}_{y_r}^2(n_1, n_2) - \sigma_v^2, 0\}.$$

The overall system diagram is shown in Figure 8.4–1, where we see a two-channel system, consisting of a low-frequency *local mean channel* and a high-frequency *residual channel*. The RUKF estimator of x_r can be adapted or modified by the local spatial variance $\hat{\sigma}_{x_r}^2(n_1, n_2)$ on the residual channel—i.e., by the residual signal variance estimate.

Example 8.4–1: Simple Inhomogeneous Gaussian Estimation

Here, we assume that the residual signal model is white Gaussian noise with zero mean and fixed variance $\sigma_{x_r}^2$. Since the residual observation noise is also white Gaussian, with variance σ_v^2 , we can construct the simple Wiener filter for the residual observations as

$$h(n_1, n_2) = \frac{\sigma_{x_r}^2}{\sigma_{x_r}^2 + \sigma_v^2} \delta(n_1, n_2).$$

The overall filter then becomes

$$\hat{x}(n_1, n_2) = \frac{\sigma_{x_r}^2}{\sigma_{x_r}^2 + \sigma_v^2} \hat{y}_r(n_1, n_2) + \mu_x(n_1, n_2),$$

with approximate realization through box filtering as

$$\hat{x}(n_1, n_2) = \frac{\hat{\sigma}_{x_r}^2}{\hat{\sigma}_{x_r}^2 + \sigma_v^2} \hat{y}_r(n_1, n_2) + \hat{\mu}_x(n_1, n_2).$$

This filter has been known for some time in the image processing literature, where it is referred to often as the simplest adaptive Wiener filter, discovered by Wallis [14]. This is essentially the `wiener2` function in the Image Processing Toolbox of MATLAB (R2010a).

Inhomogeneous Estimation with RUKF

Now for a given fixed residual model (8.4–1), there is a multiplying constant A that relates the model residual noise variance $\sigma_{w_r}^2$ to the residual signal variance $\sigma_{x_r}^2$. The residual model itself can be determined from a least-squares prediction on similar, but noise-free, residual image data. We can then construct a space-variant RUKF on these data, where the residual image model is constant except for changes in the signal model noise variance. Instead of actually running the space-variant RUKF error covariance equations, we simplify the problem by quantizing the estimate $\hat{\sigma}_{x_r}^2$ into three representative values and then use a steady-state RUKF for each of these three, just switching between them as the estimated model noise variance changes [8].

Example 8.4–2: Inhomogeneous RUKF Estimate

Here, we observe the Lena image in 10-dB white Gaussian noise, but we assume an inhomogeneous Gaussian model. The original image and input 10-dB noisy image are the same as in Figure 8.3–2. Four estimates are then shown in Figure 8.4–2: (a) is an LSI estimate, (b) is the simple Wallis filter estimate, (c) is the residual RUKF estimate, and (d) is a normalized RUKF estimate that is defined by [8]. We can notice that all three adaptive or inhomogeneous estimates look better than the LSI RUKF estimate, even though the Wallis estimate has the lowest ISNR, as seen in Table 8.4–1.

Table 8.4–1 Obtained ISNRs for Five Filter Types

Input SNR	10 dB	3 dB
LSI RUKF	4.9	8.6
Simple Wiener (Wallis)	4.3	7.8
3-Gain residual RUKF	6.0	9.3
3-Gain normalized RUKF	5.2	9.1

An inhomogeneous RUKF restoration was given in Example 8.3–2.



FIGURE 8.4-2

Various inhomogeneous Gaussian estimates: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. (a) LSI, (b) Wallis filter, (c) residual RUKF, and (d) normalized RUKF.

8.5 ESTIMATION IN THE SUBBAND/WAVELET DOMAIN

Generalizing the preceding two-channel system, consider that first a subband/wavelet decomposition is performed on the input image. We would have one low-frequency channel, similar to the local mean channel, and then many intermediate and high-frequency subband channels. As before, we can, in a suboptimal fashion, perform estimates of the signal content on each subband separately. Finally, we

create an estimate in the spatial domain by performing the corresponding inverse subband/wavelet transform (SWT). A basic contribution in this area was by Donoho [15] who applied noise-thresholding to the subband/wavelet coefficients. Later, Zhang et al. [16] introduced simple adaptive Wiener filtering for use in this *wavelet image denoising*.

In these works, it has been found useful to employ the so-called *overcomplete* subband/wavelet transform (OCSWT), which results when the decimators are omitted in order to completely remove any shift-varying effects. Effectively there are four phases computed at the first splitting stage; this then progresses geometrically down the subband/wavelet tree. In the reconstruction phase, an average over these phases must be done to arrive at the overall estimate in the image domain. Both hard and soft thresholds have been considered, the basic idea being that small coefficients are most likely due to the assumed white observation noise. If the wavelet coefficient, or subband value, is greater than a well-determined threshold, it is likely to be “signal” rather than “noise.” The estimation of the threshold is done based on the assumed noise statistics. While only constant thresholds are considered in [15], a spatially adaptive thresholding is considered in [17].

Example 8.5–1: SWT Denoising

Here, we look at estimates obtained by thresholding in the SWT domain. We start with the 256×256 monochrome Lena image and then add Gaussian white noise to make the PSNR⁶ = 22 dB. Then we input this noisy image into a five-stage SWT using an eight-tap orthogonal wavelet filter due to Daubechies. Calling the SWT domain image $y(n_1, n_2)$, we write the thresholding operation as

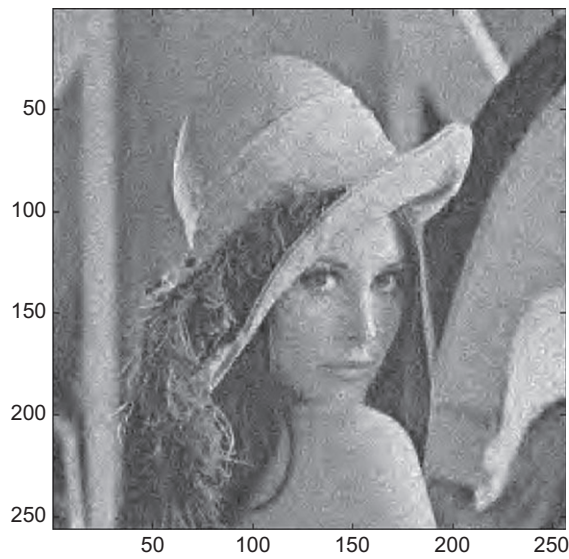
$$\hat{y} = \begin{cases} 0, & |y| < t, \\ y, & |y| \geq t, \end{cases}$$

where the noise-threshold value was taken as $t = 40$ on the 8-bit image scale $[0, 256]$. Using this *hard threshold*, we obtain the image in Figure 8.5–1. We also tried the soft threshold given as

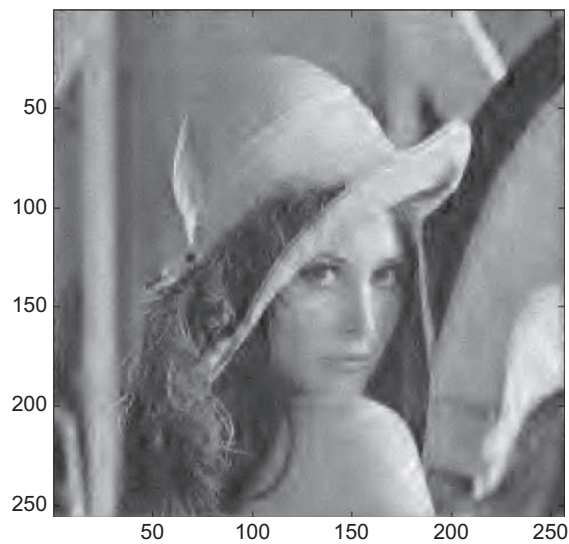
$$\hat{y} = \begin{cases} \text{sgn}(y)(|y| - t), & |y| < t, \\ y, & |y| \geq t, \end{cases}$$

with the result shown in Figure 8.5–2. Using the OCSWT and hard threshold, we obtain Figure 8.5–3. Finally, the result of soft threshold in the OCSWT domain is shown in Figure 8.5–4.

⁶PSNR stands for peak SNR and is defined as $20\log_{10}(255/\text{MSE})$ for 8-bit images. This widely used objective measure of image quality is closely related to SNR, but with 255 substituted for the image standard deviation. Note that 255 is the largest 8-bit image intensity, the values being 0 to $256 - 1$. For images with higher bit depths, the peak will be different, e.g., for 10 bits the peak value changes from 255 to $2^{10} - 1 = 1023$.

**FIGURE 8.5-1**

Estimate using hard threshold in the SWT domain.

**FIGURE 8.5-2**

Estimate using soft threshold in the SWT domain.

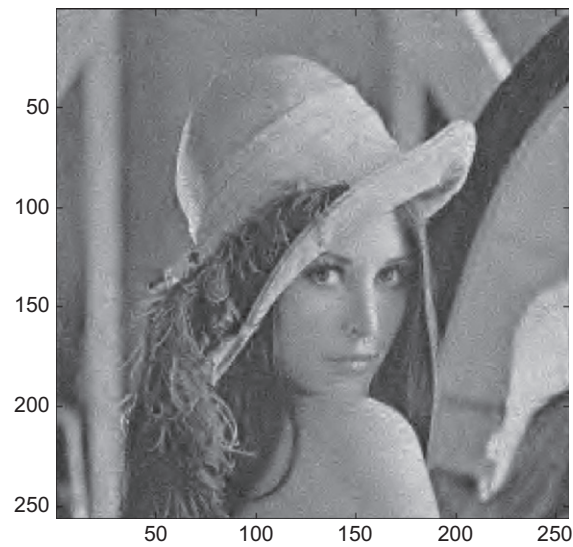


FIGURE 8.5-3

Hard threshold $\tau = 40$ in the OCSWT domain.

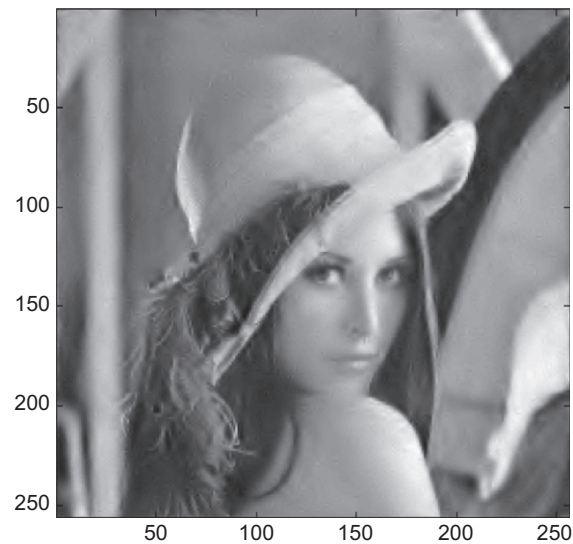


FIGURE 8.5-4

Estimate using soft threshold in the OCSWT domain.

For the SWT, the choice of this threshold $t = 40$ resulted in output PSNR = 25 dB in the case of hard thresholding, while the soft threshold gave 26.1 dB. For the OCSWT, we obtained 27.5 dB for the hard threshold and 27 dB for the soft threshold. In the case of the OCSWT, there is no inverse transform because it is overcomplete, so a least-squares inverse was used. ■

Besides these simple noise-thresholding operations, we can perform the Wiener or Kalman filter in the subband domain. An example of this appears at the end of [Section 8.7](#) on image and blur model identification.

8.6 BAYESIAN AND MAXIMUM *A POSTERIORI* ESTIMATION

All the estimates considered thus far were Bayesian, meaning they optimized over not only the *a posteriori* observations but also jointly over the *a priori* image model in order to obtain an estimate that was close to minimum MSE. We mainly used Gaussian models that resulted in linear and sometimes space-variant linear estimates. In this section, we consider a nonlinear Bayesian image model that must rely on a global iterative recursion for its solution. We use a so-called Gibbs model for a conditionally Gaussian random field, where the conditioning is on a lower level (unobserved) line field that specifies the location of edge discontinuities in the upper level (observed) Gaussian field. The line field is used to model edgelinear features in the image. It permits the image model to have edges, across which there is low correlation, as well as smooth regions of high correlation. The resulting estimates then tend to retain the edges that would otherwise be smoothed over by an LSI filter. While this method can result in much higher quality estimations and restorations, it is by far the most computationally demanding of the estimators presented thus far. The Gauss-Markov and compound Gauss-Markov models are introduced next.

Gauss-Markov Image Model

We start with the following noncausal, homogeneous Gaussian image model,

$$x(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_c} c(k_1, k_2) x(n_1 - k_1, n_2 - k_2) + u(n_1, n_2), \quad (8.6-1)$$

where the coefficient support \mathcal{R}_c is a noncausal neighborhood region centered on $\mathbf{0}$, and the image model noise u is Gaussian and zero mean, with correlation given as

$$R_u(m_1, m_2) = \begin{cases} \sigma_u^2, & (m_1, m_2) = \mathbf{0}, \\ -c(m_1, m_2) \sigma_u^2, & (m_1, m_2) \in \mathcal{R}_c, \\ 0, & \text{elsewhere.} \end{cases} \quad (8.6-2)$$

This image model is then Gaussian and Markov in the 2-D or spatial sense [18]. The image model coefficients $c_{k_1, k_2} \triangleq c(k_1, k_2)$ provide an MMSE interpolation \hat{x} based on the neighbor values in \mathcal{R}_c ,

$$\hat{x}(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}_c} c_{k_1, k_2} x(n_1 - k_1, n_2 - k_2),$$

and the image model noise u is then the resulting interpolation error.

Note that u is not a white noise, but is colored and with a finite correlation support of small size \mathcal{R}_c , beyond which it is uncorrelated, and because it is Gaussian, also independent. This noncausal notion of Markov is somewhat different from the NSHP causal Markov we have seen earlier. The *noncausal Markov random field* is defined by

$$f_x(x(\mathbf{n}) | \text{all other } x) = f_x(x(\mathbf{n}) | x(\mathbf{n} - \mathbf{k}), \mathbf{k} \in \mathcal{R}_c),$$

where \mathcal{R}_c is a small neighborhood region centered on, but not including, $\mathbf{0}$. If the random field is Markov in this noncausal or spatial sense, then the best estimate of $x(\mathbf{n})$ can be obtained using only those values of x that are neighbors in the sense of belonging to the set $\{x | x(\mathbf{n} - \mathbf{k}), \mathbf{k} \in \mathcal{R}_c\}$. A helpful diagram illustrating the noncausal Markov concept is Figure 8.6–1, which shows a central region \mathcal{G}^+ where the random field x is conditionally independent of its values on the outside region \mathcal{G}^- , given its values on a boundary region $\partial\mathcal{G}$ of the minimum width given by the neighborhood region \mathcal{R}_c . The “width” of \mathcal{R}_c then gives the “order” of the Markov field. If we would want to compare this noncausal Markov concept to that of the NSHP causal Markov random fields considered earlier, we can think of the boundary region $\partial\mathcal{G}$ as being stretched out to infinity in such a way that we get the situation depicted in Figure 8.6–2a, where the boundary region $\partial\mathcal{G}$ then becomes just the global state vector support in the case of the NSHP Markov model for the 2-D Kalman filter. Another possibility is shown in Figure 8.6–2b, which denotes a vector concept of causality wherein scanning proceeds a full line at a time. Here, the *present* is the whole line $x(n)$, in an obvious notation. In all three concepts, region \mathcal{G}^- is called the *past*, boundary region $\partial\mathcal{G}$ is called the *present*, and region \mathcal{G}^+ is called the *future*. While the latter two are consistent with some form of causal or sequential processing in the estimator, the noncausal Markov, illustrated in Figure 8.6–1, is not, and thus requires iterative processing for its estimator solution.

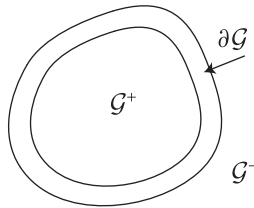


FIGURE 8.6–1

Illustration of dependency regions for the noncausal Markov field.

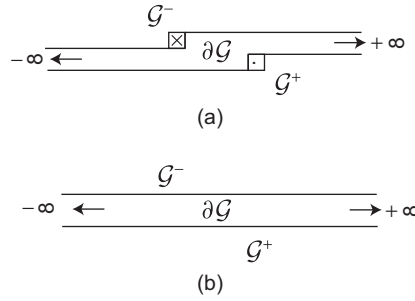
**FIGURE 8.6-2**

Illustration of two causal Markov concepts.

Turning to the Fourier transform of the correlation of the image model noise u , we see that the PSD of the model noise random field u is

$$S_u(\omega_1, \omega_2) = \sigma_u^2 \left(1 - \sum_{(k_1, k_2) \in \mathcal{R}_c} c(k_1, k_2) \exp -j(k_1 \omega_1 + k_2 \omega_2) \right)$$

and that the PSD of the image random field x is then given as, via application of (8.1-3),

$$S_x(\omega_1, \omega_2) = \frac{\sigma_u^2}{\left[1 - \sum_{(k_1, k_2) \in \mathcal{R}_c} c(k_1, k_2) \exp -j(k_1 \omega_1 + k_2 \omega_2) \right]}. \quad (8.6-3)$$

In order to write the pdf of the Markov random field, we turn to the theory of *Gibbs distributions*, as shown in [19], where it was established that the unconditional joint pdf of a Markov random field x can be expressed as

$$f_x(\mathbf{X}) = K \exp -U_x(\mathbf{X}), \quad (8.6-4)$$

where the matrix \mathbf{X} denotes x restricted to the finite region \mathcal{X} , $U_x(\mathbf{X})$ is an *energy function* defined in terms of *potential functions* V as

$$U_x(\mathbf{X}) \triangleq \sum_{c_n \in C_n} V_{c_n}(\mathbf{X}), \quad (8.6-5)$$

where C_n denotes a *clique system* in the finite region \mathcal{X} , and K is a normalizing constant. Here, a *clique* is a link between $x(\mathbf{n})$ and its immediate neighbors in the neighborhood region \mathcal{R}_c . An example of these concepts is given next.

Example 8.6-1: First-Order Clique System

Let the homogeneous random field x be noncausal Markov with neighborhood region \mathcal{R}_c of the form

$$\mathcal{R}_c = \{(1, 0), (0, 1), (-1, 0), (0, -1)\},$$

which we can call first-order noncausal Markov. Then the conditional pdf of $x(\mathbf{n}) = x(n_1, n_2)$ can be given as

$$f_x(x(\mathbf{n}) | \text{all other } x) = f_x(x(\mathbf{n}) | \{x(\mathbf{n} - (1, 0)), x(\mathbf{n} - (0, 1)), x(\mathbf{n} - (-1, 0)), x(\mathbf{n} - (0, -1))\}),$$

and the joint pdf of x over finite region \mathcal{X} can be written in terms of energy function (8.6–5), with potentials V_{c_n} of the form

$$V_{c_n} = \frac{x^2(\mathbf{n})}{2\sigma_u^2} \quad \text{or} \quad V_{c_n} = -\frac{c(\mathbf{k})x(\mathbf{n})x(\mathbf{n} - \mathbf{k})}{2\sigma_u^2} \quad \text{for each } \mathbf{k} \in \mathcal{R}_c.$$

Compound Gauss-Markov Image Model

The partial difference equation model for the compound Gauss-Markov (CGM) random field is given as

$$x(n_1, n_2) = \sum_{(k_1, k_2) \in \mathcal{R}} c^{I(n_1, n_2)}(k_1, k_2) x(n_1 - k_1, n_2 - k_2) + u^{I(n_1, n_2)}(n_1, n_2),$$

where $I(n_1, n_2)$ is a vector of four nearest neighbors from a line field $I(n_1, n_2)$, which originated in Geman and Geman [20]. This line field exists on an interpixel grid and takes on two values to indicate whether a *bond* is in-place or broken between two pixels, both horizontal and vertical neighbors. An example of this concept is shown in Figure 8.6–3, which shows a portion of a fictitious image with an edge going downwards, as modeled by a line field. The black lines indicate broken bonds ($l = 0$).

The model interpolation coefficients $c^{I(n_1, n_2)}(k_1, k_2)$ vary based on in this case the 4 nearest neighbor line field values, as captured in the line field vector $I(n_1, n_2)$. The image model does not attempt to smooth over an edge if the line field indicates a broken bond. For the line field potentials V , we have used the model suggested in [20] and shown in Figure 8.6–4, where a broken bond is indicated by a line, an in-place bond by no line, and the pixel locations with large dots. Note that only one rotation of the indicated neighbor pattern is shown. We see that the potentials favor (with $V = 0.0$) bonds being in-place, with the next most favorable situation

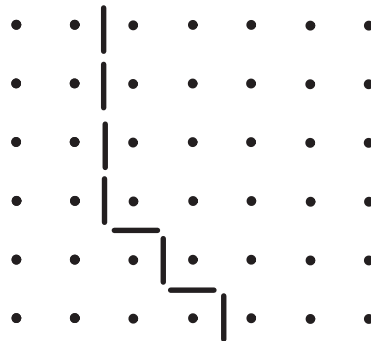
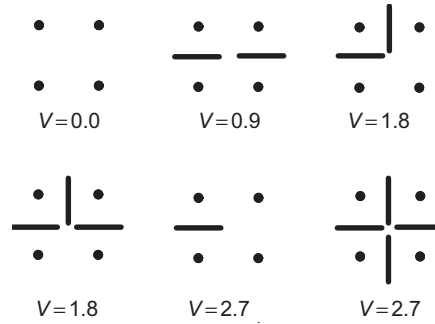


FIGURE 8.6–3

Example of a line field modeling edge in a portion of a fictitious image.

**FIGURE 8.6-4**

Line field potential V values for indicated nearest neighbors (black lines indicate bond broken).

being a horizontal or vertical edge (with $V = 0.9$), and so on to less often occurring configurations.

Then the overall Gibbs probability mass function (pmf) for the line field can be written as

$$p(\mathbf{L}) = K_1 \exp -U_l(\mathbf{L}),$$

with

$$U_l(\mathbf{L}) \triangleq \sum_{c_l \in C_l} V_{c_l}(\mathbf{L}),$$

with \mathbf{L} denoting a matrix of all line field values for the image. The overall joint pdf/pmf for the CGM field over a finite region can then be written as

$$f(\mathbf{X}, \mathbf{L}) = K_2 \exp -(U_x(\mathbf{X}) + U_l(\mathbf{L})).$$

We should note that \mathbf{L} needs about twice as many points in it as \mathbf{X} , since there is a link to the right and below each observed pixel, except for the last row and last column.

Simulated Annealing

In the simulated annealing (SA) method, a *temperature* parameter T is added to the preceding joint pdf, so that it becomes

$$f(\mathbf{X}, \mathbf{L}) = K_2 \exp -\frac{1}{T}(U_x(\mathbf{X}) + U_l(\mathbf{L})).$$

At $T = 1$, we have the correct joint-mixed pdf, but as we slowly lower $T \searrow 0$, the global maximum moves relatively higher than the set of local maxima, and this property can be used in iteratively locating the *maximum a posteriori* (MAP) estimate

$$(\hat{\mathbf{X}}, \hat{\mathbf{L}}) \triangleq \arg \max_{\mathbf{X}, \mathbf{L}} f(\mathbf{X}, \mathbf{L} | \mathbf{R}).$$

This iterative solution method, called *simulated annealing* (SA), is developed in Jeng and Woods [21]; it alternates between pixel update and line field update, as it

completes passes through the received data \mathbf{R} . At the end of each complete pass, the temperature T is reduced a small amount in an effort to eventually *freeze* the process at the joint MAP estimates $(\hat{\mathbf{X}}, \hat{\mathbf{L}})$. A key aspect of SA is determining the conditional pdf's for each pixel and line field location given the observations \mathbf{R} and all the other pixel and line field values. The following conditional pdf is derived in [21]:

$$f_x(n_1, n_2) = K_3 \exp - \left[\frac{(x(n_1, n_2) - \sum_{(k_1, k_2) \in \mathcal{R}} c^{I(n_1, n_2)}(k_1, k_2) x(n_1 - k_1, n_2 - k_2))^2}{2T\sigma_{u, I(n_1, n_2)}^2} - \frac{\sum_{(k_1, k_2) \in \mathcal{R}_h} (x(n_1 + k_1, n_2 + k_2) - \sum_{(m_1, m_2) \in \mathcal{R}_h} h(m_1, m_2) x(n_1 + k_1 - m_1, n_2 + k_2 - m_2))^2}{2T\sigma_v^2} \right],$$

and for each line field location (n_1, n_2) between pixels (i_1, i_2) and (j_1, j_2) ⁷, both vertically and horizontally, the conditional updating pmf

$$p_l(n_1, n_2) = K_4 \exp - \left[\frac{x^2(i_1, i_2)}{2T\sigma_{u, I(i_1, i_2)}^2} - \frac{c^{I(j_1, j_2)}(i_1 - j_1, i_2 - j_2) x(i_1, i_2) x(j_1, j_2)}{T\sigma_{u, I(i_1, i_2)} \sigma_{u, I(j_1, j_2)}} + \frac{x^2(j_1, j_2)}{2T\sigma_{u, I(j_1, j_2)}^2} + \frac{1}{T} \sum_{(n_1, n_2) \in C_l} V_{cl}(\mathbf{L}) \right],$$

where K_3 and K_4 are normalizing constants. These conditional distributions are *sampled* as we go through the sweeps, meaning that at each location \mathbf{n} , the previous estimates of \mathbf{l} and \mathbf{x} are replaced by values drawn from these conditional distributions. In evaluating these conditional distributions, the latest available estimates of the other \mathbf{l} or \mathbf{x} values are always used. This procedure is referred to as a *Gibbs sampler* [20]. The number of iterations that must be done is a function of how fast the temperature decreases. Theoretical proofs of convergence [20, 21] require a very slow logarithmic decrease of temperature with the sweep number n ; that is,

$$T = C / \log(1 + n).$$

However, in practice a faster decrease is used. A typical number of sweeps necessary was experimentally found to be in the 100s.

Example 8.6–2: SA for Image Estimation

In [21], we modeled the 256×256 monochrome Lena image by a CGM model with order 1×1 and added white Gaussian noise to achieve an input SNR = 10 dB. The global mean was then subtracted before processing. For comparison purposes, we first show the Wiener filter result in Figure 8.6–5. There is substantial noise reduction but also visible blurring.

⁷We should note that the line field \mathbf{l} is on the interpixel grid, with about twice the number of points as pixels in image \mathbf{x} , so that $x(\mathbf{n})$ is not the same location as $l(\mathbf{n})$.

The SA result, after 200 iterations, is shown in Figure 8.6–6, where we see a much stronger noise reduction combined with an almost strict preservation of important visible edges.



FIGURE 8.6–5

Example of Wiener filtering for the Gauss-Markov model at input SNR = 10 dB.



FIGURE 8.6–6

SA estimate for a CGM model at input SNR = 10 dB.

Example 8.6–3: SA for Image Restoration

In [21], we blurred the Lena image with a 5×5 uniform blur function and then added a small amount of white Gaussian noise to achieve a $\text{BSNR} = 40$ dB, with the result shown in Figure 8.6–7. The global mean was then subtracted before processing. We restored this noisy and blurred image with a Wiener filter to produce the result shown in Figure 8.6–8,

**FIGURE 8.6–7**

Input blurred image as $\text{BSNR} = 40$ dB.

**FIGURE 8.6–8**

Blur restoration via the Wiener filter.

**FIGURE 8.6–9**

Blur restoration via SA.

which is considerably sharper but contains some ringing artifacts. Then we processed the noisy blurred image with SA at 200 iterations to produce the image shown in Figure 8.6–9, which shows a sharper result and with reduced image ringing artifacts. ■

A nonsymmetric half-plane (NSHP) causal CGM image model, called *doubly stochastic Gaussian* (DSG), was also formulated [21], and a sequential noniterative solution method for an approximate MMSE was developed using a so-called *M-algorithm*. Experimental results are provided in [21] and also show considerable improvement over the LSI Wiener and RUKF filters. The *M*-algorithms are named for their property of following *M* paths of chosen directional NSHP causal image models through the data. A relatively small number of paths sufficed for the examples in [21].

Extensions of Simulated Annealing

The preceding SA restoration algorithm is very adaptable to massive parallel processing [22]. Also in addition to MAP estimates, the Gibbs sampler can be used to generate approximate MMSE estimates via

$$\hat{X}_{MS} = \frac{1}{K} \sum_{k=1}^K X_k,$$

where the X_k are the result of the k th run, all done with temperature $T = 1$.

Example 8.6–4: Parallel SA

The following estimates were generated in a parallel implementation of SA. Figure 8.6–10a shows the parallel MAP restoration result for the previous 5×5 blur with $\text{BSNR} = 40$ dB.

**FIGURE 8.6–10**

Parallel SA restoration results: (a) MAP estimate. (b) MMSE estimate.

Figure 8.6–10b is the MMSE estimate computed by the Gibbs sampler with constant temperature $T = 1$. Both images are the result of 200 iterations.

The SNR improvement was $\text{ISNR} = 6.33 \text{ dB}$ for the MAP estimate and $\text{ISNR} = 6.85$ for the MMSE estimate. ■

An extension to the CGM approach with improved convergence properties for severely blurred images appears in [23]. A variation on the line field prior is the CGM with edge-directed prior contained in [24].

8.7 IMAGE IDENTIFICATION AND RESTORATION

Here, we consider the additional practical need in many applications to estimate the signal model parameters, as well as the parameters of the observation, including the blur function in the restoration case. This can be a daunting problem, with no solution admitted in some cases. Fortunately, this *identification* or model parameter estimation problem can be solved in many practical cases. Here, we present a method of combined identification and restoration using the *expectation-maximization* algorithm.

Expectation-Maximization Algorithm Approach

We will use the 2-D vector/4-D matrix notation of problem 20 in Chapter 4 and follow the development of Lagendijk et al. [25]. First, we establish an AR signal model in 2-D/4-D matrix/vector form as⁸

$$X = CX + W, \quad (8.7-1)$$

⁸The reader should note that the matrix-vector equations are the same as for a stacked vector and block matrix. The difference is internal to the denoted matrix and vector objects.

and then write the observations as

$$\mathbf{Y} = \mathcal{D}\mathbf{X} + \mathbf{V}, \quad (8.7-2)$$

with the signal model noise \mathbf{W} and the observation noise \mathbf{V} both Gaussian, zero mean, and independent of one another, with variances σ_w^2 and σ_v^2 , respectively. We assume that the 4-D matrices \mathcal{C} and \mathcal{D} are parameterized by the filter coefficients $c(k_1, k_2)$ and $d(k_1, k_2)$, respectively. Each such filter is finite order and restricted to an appropriate region, that is to say, an NSHP region of support for the model coefficients c , and typically a rectangular region centered on the origin for the blur model coefficients d . All model parameters can be conveniently written together in the vector parameter

$$\Theta \triangleq \{c(k_1, k_2), d(k_1, k_2), \sigma_w^2, \sigma_v^2\}.$$

To ensure uniqueness, we assume that the blurring coefficients $d(k_1, k_2)$ are normalized to sum to one,

$$\sum_{k_1, k_2} d(k_1, k_2) = 1, \quad (8.7-3)$$

which is often reasonable when working in the intensity domain, where this represents a conservation of power, as would be approximately true with a lens.

This parameter vector is unknown and must be estimated from the noisy data \mathbf{Y} , and we seek the maximum-likelihood (ML) estimate [1] of this unknown but nonrandom parameter,

$$\hat{\Theta}_{ML} \triangleq \arg \max_{\Theta} \{\log f_Y(\mathbf{Y}; \Theta)\}, \quad (8.7-4)$$

where f_Y is the pdf of the noisy observation vector \mathbf{Y} . We note that this ML estimate is the choice of the parameter Θ that makes the given observation most probable for the given family of densities f_Y , and thus seems reasonable. In fact, the ML estimate of unknown parameters is much studied and used in estimation and detection applications.

Since \mathbf{X} and \mathbf{Y} are jointly Gaussian, we can write

$$f_X(\mathbf{X}) = \sqrt{\frac{\det(\mathcal{I} - \mathcal{C})^2}{(2\pi)^{N^2} \det \mathcal{Q}_W}} \exp - \frac{1}{2} \left\{ \mathbf{X}^T (\mathcal{I} - \mathcal{C})^t \mathcal{Q}_W^{-1} (\mathcal{I} - \mathcal{C}) \mathbf{X} \right\}$$

and

$$f_{Y|X}(\mathbf{Y}|\mathbf{X}) = \frac{1}{\sqrt{(2\pi)^{N^2} \det \mathcal{Q}_V}} \exp - \frac{1}{2} \left\{ (\mathbf{Y} - \mathcal{D}\mathbf{X})^t \mathcal{Q}_V^{-1} (\mathbf{Y} - \mathcal{D}\mathbf{X}) \right\}.$$

Combining (8.7-1) and (8.7-2), we have

$$\mathbf{Y} = \mathcal{D}(\mathcal{I} - \mathcal{C})^{-1} \mathbf{W} + \mathbf{V},$$

with covariance matrix

$$\mathcal{K}_{YY} = \mathcal{D}(\mathcal{I} - \mathcal{C})^{-1} \mathcal{Q}_W (\mathcal{I} - \mathcal{C})^{-1} \mathcal{D}^t + \mathcal{Q}_V,$$

so the needed ML estimate of the parameter vector Θ can be expressed as

$$\hat{\Theta}_{ML} = \arg \max_{\Theta} \left\{ -\log(\det(\mathcal{K}_{YY})) - \mathbf{Y}^t \mathcal{K}_{YY}^{-1} \mathbf{Y} \right\}, \quad (8.7-5)$$

which is unfortunately highly nonlinear and not amenable to closed-form solution. Further, there are usually several to many local maxima to worry about. Thus we take an alternative approach and arrive at the *expectation maximization* (EM) algorithm, an iterative method that converges to the local optimal points of this equation.

The EM method (see also Section 9.4 in [1]) talks about so-called *complete* and *incomplete* data. In our case, the complete data are $\{X, Y\}$, and the incomplete data are just $\{Y\}$. Given the complete data, we can easily solve for $\hat{\Theta}_{ML}$; specifically, we would obtain $c(k_1, k_2)$ and σ_w^2 as the solution to a 2-D linear prediction problem, expressed as a 2-D Normal equation (see Section 8.1). Then the parameters $d(k_1, k_2)$ and σ_v^2 would be obtained via

$$\hat{d}(k_1, k_2), \hat{\sigma}_v^2 = \arg \max f(Y|X) \sim N(\mathcal{D}X, \mathcal{Q}_V),$$

which is easily solved via classical system theory. Note that this follows only because the pdf of the complete data separates, i.e.,

$$f(X, Y) = f(X)f(Y|X),$$

with the image model parameters only affecting the first factor, and the observation parameters only affecting the second factor. The EM algorithm effectively converts the highly nonlinear problem (8.7–5) into a sequence of problems that can be solved as simply as if we had the complete data. In fact, it is crucial in an EM problem to formulate it, if possible, so that the ML estimate is easy to solve given the complete data. Fortunately, this is the case here.

EM Algorithm

Start at $k = 0$ with an initial guess $\hat{\Theta}^0$. Then alternate the following E-step and M-step until convergence:

$$\begin{aligned} \text{E-step; } \mathcal{L}(\Theta; \hat{\Theta}^{(k)}) &\triangleq E[\log f(X, Y; \Theta) | Y; \hat{\Theta}^{(k)}] \\ &= \int \log f(X, Y; \Theta) f(X|Y; \hat{\Theta}^{(k)}) dX, \\ \text{M-step; } \hat{\Theta}^{(k+1)} &= \arg \max_{\Theta} \mathcal{L}(\Theta; \hat{\Theta}^{(k)}). \end{aligned}$$

It is proven [26] that this algorithm will monotonically improve the likelihood of the estimate $\hat{\Theta}^{(k)}$ and so result in a local optimum of the objective function given in (8.7–4).

We now proceed with the calculation of $\mathcal{L}(\Theta; \hat{\Theta}^{(k)})$ by noting that

$$\begin{aligned} f(X, Y; \Theta) &= f(Y|X)f(X; \Theta) \quad \text{and} \\ f(X|Y; \hat{\Theta}^{(k)}) &= \frac{f(X, Y; \hat{\Theta}^{(k)})}{f(Y; \hat{\Theta}^{(k)})} \\ &= \frac{1}{\sqrt{(2\pi)^{N^2} \det \hat{\mathcal{K}}_{EE}^{(k)}}} \exp -\frac{1}{2} \left\{ (X - \hat{X}^{(k)})^t \left(\hat{\mathcal{K}}_{EE}^{(k)} \right)^{-1} (X - \hat{X}^{(k)}) \right\}, \end{aligned}$$

where $\hat{\mathbf{X}}^{(k)}$ and $\hat{\mathbf{K}}_{EE}^{(k)}$ are, respectively, the conditional mean and conditional variance matrices of \mathbf{X} at the k th iteration. Here, we have

$$\begin{aligned}\hat{\mathbf{X}}^{(k)} &\triangleq E[\mathbf{X}|\mathbf{Y}; \hat{\boldsymbol{\Theta}}^{(k)}] = \hat{\mathbf{K}}_{EE}^{(k)} \mathbf{D}^t \mathbf{Q}_v^{-1} \mathbf{Y}, \quad \text{with estimated error covariance} \\ \hat{\mathbf{K}}_{EE}^{(k)} &\triangleq \text{cov}[\mathbf{X}|\mathbf{Y}; \hat{\boldsymbol{\Theta}}^{(k)}] = \{(\mathbf{I} - \mathbf{C})^t \mathbf{Q}_w^{-1} (\mathbf{I} - \mathbf{C})^t + \mathbf{D}^t \mathbf{Q}_v^{-1} \mathbf{D}\}^{-1}.\end{aligned}$$

Thus the spatial Wiener filter designed with the parameters $\hat{\boldsymbol{\Theta}}^{(k)}$ will give us also the likelihood function $\mathcal{L}(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}^{(k)})$ to be maximized in the M-step.

Turning to the M-step, we can express $\mathcal{L}(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}^{(k)})$ as the sum of terms

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}^{(k)}) &= c - N^2 \log(\sigma_w^2 \sigma_v^2) + \log \det(\mathbf{I} - \mathbf{C})^2 \\ &\quad - \frac{1}{\sigma_v^2} \mathbf{Y}^T \mathbf{Y} + \frac{2}{\sigma_v^2} \text{tr}(\mathbf{D} \hat{\mathbf{K}}_{XY}) - \frac{1}{\sigma_v^2} \text{tr}(\mathbf{D} \hat{\mathbf{K}}_{XX} \mathbf{D}^t) \\ &\quad - \frac{1}{\sigma_w^2} \text{tr}\{(\mathbf{I} - \mathbf{C}) \hat{\mathbf{K}}_{XX} (\mathbf{I} - \mathbf{C})^t\},\end{aligned}\tag{8.7-6}$$

where c is a constant and

$$\hat{\mathbf{K}}_{XY}^{(k)} \triangleq E[\mathbf{X} \mathbf{Y}^t | \mathbf{Y}; \hat{\boldsymbol{\Theta}}^{(k)}] = \hat{\mathbf{X}}^{(k)} \mathbf{Y}^t \tag{8.7-7}$$

and

$$\hat{\mathbf{K}}_{XX}^{(k)} \triangleq E[\mathbf{X} \mathbf{X}^t | \mathbf{Y}; \hat{\boldsymbol{\Theta}}^{(k)}] = \hat{\mathbf{K}}_{EE}^{(k)} + \hat{\mathbf{X}}^{(k)} \hat{\mathbf{X}}^{(k)t}. \tag{8.7-8}$$

Notice that (8.7-6) separates into two parts, one of which depends on the signal model parameters and one of which only depends on the observation model parameters; thus the maximization of $\mathcal{L}(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}^{(k)})$ can be separated into these two distinct parts. The first part involving image model identification becomes

$$\arg \max_{c(k_1, k_2), \sigma_w^2} \left\{ \log \det(\mathbf{I} - \mathbf{C})^2 - N^2 \log \sigma_w^2 - \frac{1}{\sigma_w^2} \text{tr}\{(\mathbf{I} - \mathbf{C}) \hat{\mathbf{K}}_{XX}^{(k)} (\mathbf{I} - \mathbf{C})^t\} \right\},$$

and the first term can be deleted since an NSHP causal model must satisfy $\det(\mathbf{I} - \mathbf{C}) = 1$. The resulting simplified equation is quadratic in the image model coefficients $c(k_1, k_2)$ and is solved by the 2-D Normal equations. The remaining part of (8.7-6) becomes

$$\arg \max_{d(k_1, k_2), \sigma_v^2} \left\{ -N^2 \log \sigma_v^2 - \frac{1}{\sigma_v^2} \mathbf{Y}^T \mathbf{Y} + \frac{2}{\sigma_v^2} \text{tr}(\mathbf{D} \hat{\mathbf{K}}_{XY}) - \frac{1}{\sigma_v^2} \text{tr}(\mathbf{D} \hat{\mathbf{K}}_{XX}^{(k)} \mathbf{D}^t) \right\},$$

which is quadratic in the blur coefficients $d(k_1, k_2)$ and thus easy to maximize for any σ_v^2 . Then the estimate of σ_v^2 can be found.

If the preceding equations are circulant approximated, then due to the assumption of constant coefficients and noise variances, we can write scalar equations for each

of these two identification problems. We get from (8.7-7) and (8.7-8),

$$\hat{K}_{XX}^{(k)}(m_1, m_2) = \hat{K}_{EE}^{(k)}(m_1, m_2) + \frac{1}{N^2} \sum_{k_1, k_2} \hat{x}^{(k)}(k_1, k_2) \hat{x}^{(k)}(k_1 - m_1, k_2 - m_2) \quad (8.7-9)$$

$$\hat{K}_{XY}^{(k)}(m_1, m_2) = \frac{1}{N^2} \sum_{k_1, k_2} \hat{x}^{(k)}(k_1, k_2) y(k_1 - m_1, k_2 - m_2), \quad (8.7-10)$$

where we have summed over all the element pairs with shift distance (m_1, m_2) . Then for the image model identification, we get the Normal equations

$$\hat{K}_{XX}^{(k)}(m_1, m_2) = \sum_{\mathcal{S}_c} \hat{c}(k_1, k_2) \hat{K}_{XX}^{(k)}(m_1 - k_1, m_2 - k_2), \quad \text{for all } (m_1, m_2) \in \mathcal{S}_c,$$

$$\hat{\sigma}_w^2 = \hat{K}_{XX}^{(k)}(0, 0) - \sum_{k_1, k_2} \hat{c}(k_1, k_2) \hat{K}_{XX}^{(k)}(k_1, k_2),$$

where \mathcal{S}_c is the NSHP support of the AR signal model c .

For the blur model identification, the following equations can be derived in the homogeneous case [25]:

$$\hat{K}_{XY}^{(k)}(-m_1, -m_2) = \sum_{(k_1, k_2) \in \mathcal{S}_d} \hat{d}(k_1, k_2) \hat{K}_{XX}^{(k)}(m_1 - k_1, m_2 - k_2), \quad \text{for all } (m_1, m_2) \in \mathcal{S}_d,$$

$$\hat{\sigma}_v^2 = \frac{1}{N^2} \sum_{(n_1, n_2) = (0, 0)}^{(N-1, N-1)} y^2(n_1, n_2) - \sum_{k_1, k_2} \hat{d}(k_1, k_2) \hat{K}_{XY}^{(k)}(-k_1, -k_2),$$

where \mathcal{S}_d is the support of the FIR blur impulse response d . The resulting d values can then be normalized via (8.7-3).

EM Method in the Subband/Wavelet Domain

In [27], the preceding joint identification and restoration method was extended to work in the subband/wavelet domain. In this case, we can have separate image models in each of the subbands, where we can employ a local power level to modulate the signal power in the various subbands (equivalently wavelet coefficients). This can result in much improved clarity in the restored image, as illustrated in the set of images in 8.7-1. We note that the linear space-variant (LSV) restoration of the individual subbands works best from a visual standpoint. Its PSNR improvement is 7.3 dB while the fullband restoration only achieved 5.8-dB improvement.

Gaussian Scale Mixtures

Using an overcomplete subband/wavelet decomposition known as the steerable pyramid [28] the authors in [29] model the *individual subbands* as the product



FIGURE 8.7-1

Subband EM restoration of cameraman at BSNR = 40 dB, $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$: (a) original, (b) blurred, (c) fullband restored, (d) LL subband restored, (e) subband (LSI), and (f) subband (LSV). (from [27] © 1994 IEEE)

of a Gauss-Markov random field $u(n_1, n_2)$ and an independent positive random variable z as

$$x(n_1, n_2) = \sqrt{z}u(n_1, n_2),$$

called a *Gaussian scale mixture* (GSM). One can compute the pdf of this field on a finite rectangular region as follows. Calling the 2-D vector of the x values \mathbf{X} , we have

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{X}) &= \int f_U\left(\frac{1}{\sqrt{z}}\mathbf{X}\right)f_Z(z)dz \\ &= \int \frac{1}{(2\pi z)^{N^2/2} |\mathcal{K}_U|^{1/2}} \exp\left(-\frac{\mathbf{X}^t (z\mathcal{K}_U)^{-1} \mathbf{X}}{2}\right) f_Z(z)dz \\ &= \int \frac{1}{(2\pi z)^{N^2/2} |\mathcal{K}_X|^{1/2}} \exp\left(-\frac{\mathbf{X}^t (z\mathcal{K}_X)^{-1} \mathbf{X}}{2}\right) f_Z(z)dz. \end{aligned}$$

since we take $E[Z] = 1$, then x and u have the same covariance 4-D matrix, i.e., $\mathcal{K}_U = \mathcal{K}_X$. In [29], a separate (independent) GSM model was used for each subband scale. The motivation comes from the fact that the individual subband values tend to follow non-Gaussian *kurtotic* densities (i.e., those with sharper peaks and longer tails than Gaussian). In fact, subband/wavelet coefficients (values), DCT coefficients, and prediction errors have been found by many researchers to better fit Laplacian than Gaussian density models.

It turns out that this same equation had been suggested earlier, taking u to be exponentially distributed, as a *multivariate Laplace* distribution [30]—that is to say, as a multivariate generalization of the Laplace distribution. However, other distributions have been mentioned for z in [29], including the log Normal density (i.e., $\log z$ Gaussian). In that case, \mathbf{X} does not follow a multivariate Laplacian model, but it still can be a useful fit.

The GSM model is generalized in [31] to the *field of Gaussian scale mixtures* (FoGSM) model, which models each subband independently with the model

$$x(n_1, n_2) = \sqrt{z(n_1, n_2)} u(n_1, n_2),$$

where u and $\log z$ are now independent homogeneous Gaussian random fields. The joint pdf on a finite square region can then be written in terms of matrix \mathbf{X} as

$$f_{\mathbf{X}}(\mathbf{X}) = \int f_U(\mathbf{X}/\sqrt{\mathbf{Z}}) f_{\mathbf{Z}}(\mathbf{Z}) d\mathbf{Z},$$

where both the indicated (vector) divisions $\mathbf{X}/\sqrt{\mathbf{Z}}$ and $\log \mathbf{Z}$ defined to be computed termwise. The density $f_{\mathbf{Z}}$ is given as

$$f_{\mathbf{Z}}(\mathbf{Z}) = \frac{1}{(2\pi)^{N^2/2} \left(\prod_{i=1}^{N^2} z_i \right) |\mathcal{K}_{\log \mathbf{Z}}|^{1/2}} \exp \left(-\frac{(\log \mathbf{Z})^t (\mathcal{K}_{\log \mathbf{Z}})^{-1} (\log \mathbf{Z})}{2} \right),$$

$z_i > 0$ for all i ,

where the argument of the exp term can be written in scalar form as

$$-\frac{1}{2} \sum_{(i,j) \in [1,N]^2} \log z(i,j) \sum_{(k,l) \in [1,N]^2} c_{\log \mathbf{Z}}(k-i, l-j) \log z(k,l),$$

due to its Gauss-Markov structure, and can be implemented via 2-D convolution. Here, the coefficients are the so-called Markov-*generating kernel* values or neighborhood sets of the field $\log z(n_1, n_2)$ (see (8.6-2)). The density f_U is given as

$$f_U(\mathbf{X}/\sqrt{\mathbf{Z}}) = \frac{1}{(2\pi)^{N^2/2} \left(\prod_{i=1}^{N^2} z_i \right) |\mathcal{K}_U|^{1/2}} \exp \left(-\frac{(\mathbf{X}/\sqrt{\mathbf{Z}})^t (\mathcal{K}_U)^{-1} (\mathbf{X}/\sqrt{\mathbf{Z}})}{2} \right),$$

with argument of the exp term given as

$$\sum_{(i,j) \in [1,N]^2} x(i,j)/\sqrt{z(i,j)} \sum_{(k,l) \in [1,N]^2} c_U(k-i, l-j) x(k,l)/\sqrt{z(k,l)},$$

in terms of the generating kernel values $c_u(k,l)$ of homogeneous Gaussian random field $u(n_1, n_2)$.

Pairwise densities of subband/wavelet values and coefficients were experimentally found [31] to have a good match with Gauss-Markov members of this family using small neighborhood sets of 5×5 support; one for the random field $u(n_1, n_2)$ and one for $\log z(n_1, n_2)$. A MAP estimation method was then developed to find estimates

$$\begin{aligned}(\hat{X}, \hat{Z}) &= \arg \max_{X, Z} f(X, Z | Y) \\ &= \arg \max_{X, Z} \log f(X, Z | Y)\end{aligned}$$

for observations on the square lattice $[1, N]^2$ of the form $Y = X + V$, with V being an independent Gaussian white noise of variance σ^2 . The method is an iterative one that alternates between an estimate of X assuming $Z = \hat{Z}$ and an estimate of Z under the assumption that $X = \hat{X}$ and initialized at the GSM estimate of [29].

Example 8.7–1: FoGSM Estimate

White Gaussian noise was again added to the 256×256 Lena image to produce an SNR = 10 dB, shown in Figure 8.7–2. The global mean was then subtracted before processing. Figure 8.7–3 shows the FoGSM estimate obtained by the technique of Lyu and Simoncelli [31] with an SNR = 17.9 dB for an overall SNR improvement of ISNR = 7.9 dB.

This estimate is performed using the iterative algorithm of [31], which is implemented in the 2-D frequency domain using 2-D FFTs via a block circulant approximation of the matrices involved. The 5×5 neighborhood coefficient arrays were jointly identified as part of the iterations. (Simulation results were generously provided by Professor Siwei Lyu.)



FIGURE 8.7–2

Lena image with Gaussian noise added to yield input SNR = 10 dB.

**FIGURE 8.7–3**FoGSM estimate with $\text{ISNR} = 7.9$ dB.

8.8 NON-BAYESIAN METHODS

Here we look at some non-Bayesian methods including nonlocal means, least-squares, and total variation approaches.

Nonlocal Means

The simplest image estimator we have considered was the 3×3 box filter of Chapter 7. It shares one commonality with the much more advanced algorithms we have seen since, and that is its emphasis on using only nearby values to smooth each noisy pixel. A departure from such local estimators is to look for small blocks of noisy pixels in a larger search area that have a good match to a block centered at the current pixel, and then to average these blocks to provide the smoothed estimate. Such an approach is introduced in Buades et al. [32]. Again we take $\mathbf{Y} = \mathbf{X} + \mathbf{V}$, with \mathbf{V} and independent white Gaussian noise.

The nonlocal means estimate of [32] is formed as

$$\hat{x}(\mathbf{n}) = \sum_{i=1}^M w(\mathbf{n}, \mathbf{k}_i) y(\mathbf{k}_i),$$

where the \mathbf{k}_i are the centers of M square windows in a rather large search region \mathcal{S} surrounding \mathbf{n} , and the weights are computed via the normalized Gaussian kernel

$$w(\mathbf{n}, \mathbf{k}) \propto \exp\left(-\frac{\|\mathbf{v}(\mathbf{n}) - \mathbf{v}(\mathbf{k})\|_2^2}{h^2}\right),$$

where the $\mathbf{v}(\mathbf{k})$ are vectors of pixels in a small square window centered on pixel \mathbf{k} and the L^2 or square norm is used. The parameter h is chosen to control the weighting so that only pixel values in the center of quite similar windows will receive high weights. The normalization of weights is such that the M weights $w(\mathbf{n}, \mathbf{k}_i)$ sum to one.

Example 8.8–1: Nonlocal Means Approach

Here is an example of the nonlocal means approach of [32] where the Lena image is shown at a Gaussian white noise level of $\sigma = 20$. The search window is of size 21×21 pixels so $M = 441$. The square neighborhoods are 7×7 . The value of h was set at $h = 10\sigma$. The results are shown in Figure 8.8–1. The MSE was measured at 68. The input MSE = $\sigma^2 = 400$, so that the $\text{ISNR} = 10 \log(400/68) = 7.7 \text{ dB}$. Left is a portion of the noisy image, and right is a portion of the nonlocal (NL) image estimate. Note that the estimate is very clean and without any ringing distortion due to the fact that no spatial filtering is done.

Dabov et al. [33] expand on the nonlocal processing idea by introducing filtering on the 3-D blocks obtained by stacking the matched patches. They call this filtering *collaborative filtering* and provide results for the combination of various 3-D separable transforms together with so-called *wavelet shrinkage*, which is the same as scalar Wiener filtering on the 3-D transformed samples (see transform-domain Wiener filtering in Section 8.2). A two-step estimate is performed, where in the first step only hard thresholding of the 3-D transform coefficients is performed. In the second (final) step, Wiener filtering is done. Since the blocks partially overlap, some combining of estimates is needed. This is termed *aggregation* and is performed at the end of each step, right after the 3-D filtering. They call the overall algorithm *block-matching and 3-D filtering* (BM3D).



FIGURE 8.8–1

Nonlocal means estimator: (left) noisy segment, (right) NL estimate segment. (from Figure 5 of [32] © 2005 IEEE)

**FIGURE 8.8–2**

Collaborative filtering on the 256×256 house image: (left) noisy original, (right) estimate. (from Dabov et al. [33] © 2007 IEEE)

Example 8.8–2: BM3D estimate

The *house* image was contaminated with additive white Gaussian noise to a noise level $\sigma = 25$ to produce the noisy image shown on the left in Figure 8.8–2. The resulting BM3D estimate shown on the right. The output PSNR was given as 32.86 dB.

Least Squares and Total Variation

The general image restoration problem we have considered is of the 4-D matrix form

$$\mathbf{Y} = \mathcal{H}\mathbf{X} + \mathbf{V}, \quad (8.8-1)$$

where we typically assume that the noise \mathbf{V} is independent of the signal \mathbf{X} , and we have an assigned distribution for the image \mathbf{X} . For example, if the blur function is an LSI convolution, then

$$\mathcal{H}_{i,j,k,l} = h(i-k, j-l).$$

Our goal has been to form an estimate $\hat{\mathbf{X}}$ (linear or not) that either minimizes the MSE or maximizes the *a posteriori* probability density

$$\hat{\mathbf{X}} = \arg \max_{\mathbf{X}} f_{\mathbf{X}|\mathbf{Y}}(\mathbf{X}|\mathbf{Y}). \quad (8.8-2)$$

Now by Bayes' theorem, we have

$$\begin{aligned} f_{\mathbf{X}|\mathbf{Y}}(\mathbf{X}|\mathbf{Y}) &= f_{\mathbf{Y}|\mathbf{X}}(\mathbf{Y}|\mathbf{X})f_{\mathbf{X}}(\mathbf{X})/f_{\mathbf{Y}}(\mathbf{Y}) \\ &\sim f_{\mathbf{V}}(\mathbf{Y} - \mathcal{H}\mathbf{X})f_{\mathbf{X}}(\mathbf{X}), \end{aligned}$$

since the observation Y is fixed and known, where \sim means “is proportional to.” If we now assume that both V and X are Gaussian i.i.d. then we get

$$\log f_{X|Y}(X|Y) \sim \log f_V(Y - \mathcal{H}X) + \log f_X(X),$$

so that, assuming the means are subtracted or equivalently $\mu_X = \mu_Y = 0$, finding the $\arg \max_X$ in (8.8-2) is then equivalent to

$$\arg \min_X \left\{ \sigma_v^{-2} \|Y - \mathcal{H}X\|_2^2 + \sigma_X^{-2} \lambda \|X\|_2^2 \right\} \quad (8.8-3)$$

for some positive number λ , where the indicated norm is L^2 , given as $\|A\|_2^2 \triangleq \sum_{n_1, n_2} a^2(n_1, n_2)$.

On the other hand, a deterministic approach to image restoration given the observations (8.8-1) might well seek to find the image that minimizes the L^2 norm or “energy” of the signal (image) given some constraint on $\|Y - \mathcal{H}X\|_2^2$,

$$\arg \min_X \left\{ \|X\|_2^2 + \lambda \|Y - \mathcal{H}X\|_2^2 \right\}. \quad (8.8-4)$$

Effectively, we would look for the “smallest” signal with a close distance to the observations, both distances being measured by the L^2 norm. Taking the matrix derivative⁹ of the function in (8.8-4) with respect to the 2-D vector X , with $g \triangleq \|X\|_2^2 + \lambda \|Y - \mathcal{H}X\|_2^2$, we get

$$\begin{aligned} \nabla_X g &= \nabla_X \left(\|X\|_2^2 + \lambda \|Y - \mathcal{H}X\|_2^2 \right) \\ &= \nabla_X [X^T X + \lambda (Y - \mathcal{H}X)^T (Y - \mathcal{H}X)] \\ &= 2[X - \lambda (\mathcal{H}^T Y + \mathcal{H}^T \mathcal{H} X)]. \end{aligned}$$

Then setting the result to zero, and assuming invertability, we get the least-squares estimate \hat{X}_{LS} given as

$$\hat{X}_{LS} = (I + \lambda \mathcal{H}^T \mathcal{H})^{-1} \mathcal{H}^T Y.$$

Constrained Least Squares

A more interesting constraint would be one on the variation of X . In terms of the Laplacian operator \mathcal{L} , we may want to constrain the L^2 norm of $\mathcal{L}X$ in place of the constraint on the energy $\|X\|_2^2$. Doing so will yield the constrained least-squares problem

$$\|\mathcal{L}X\|_2^2 + \lambda \|Y - \mathcal{H}X\|_2^2,$$

⁹The matrix derivative of a 4-D matrix proceeds the same as for conventional matrix derivatives; that is, the derivative of $\mathcal{A}X$ with respect to X is \mathcal{A}^t , and the derivative of $X^t \mathcal{A}X$ is $\mathcal{A}X + \mathcal{A}^t X$.

whose minimization yields the estimate \hat{X}_{CLS} given as

$$\hat{X}_{CLS} = (\mathcal{L}'\mathcal{L} + \lambda\mathcal{H}'\mathcal{H})^{-1}\mathcal{H}'Y.$$

This method was introduced by Hunt [34], who considered a radially symmetric version of the discrete second difference operator

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

called the Laplacian filter in Chapter 7. Examples are provided in [34].

The reader may have wondered how to choose the Lagrangian parameter λ . Various estimates of suitable λ values have been obtained by statistical considerations, but most often the value of λ is set by trial and error of the experimenter to obtain best performance.

Total Variation Method

More generally, we could have exponential distributions for the noise such that the image pdf f is proportional to the exponential of the negative of an L^1 norm as

$$f \sim \exp(-\|\mathcal{L}X\|_1),$$

which would give the deterministic minimization problem

$$\|Y - \mathcal{H}X\|_2^2 + \lambda \|\mathcal{L}X\|_1. \quad (8.8-5)$$

A motivation for using this method is as follows. We have seen that the L^2 norm gives rise to excessive smoothing at edges, which after all are large excursions of the function from its mean. Thus the L^1 norm should provide less smoothing of edges, the main criticism of linear filtering of images. We could also consider the i.i.d. additive noise to be Laplacian rather than Gaussian; then the total variation problem becomes finding the value of X that minimizes

$$\|Y - \mathcal{H}X\|_1 + \lambda \|\mathcal{L}X\|_1.$$

The method is called total variation because in the 1-D case, the total variation of a function is written as $\sum_n |f(n) - f(n-1)| = \|\Delta f\|$. Unfortunately, the presence of an L^1 norm prevents an analytical solution, so recourse is made to numerical methods, and steepest descent has been used successfully. The derivatives of these quantities are readily calculated and presented in Farsiu et al. [35].

Example 8.8-3: Total Variation Estimate

We look at estimation (denoising) for the Lena 256×256 grayscale image using a total variation (TV) method based on (8.8-5) with no blurring (i.e., $\mathcal{H} = \mathcal{I}$), the identity matrix.

**FIGURE 8.8-3**

The 256×256 Lena 10-dB noisy image on the left produced the TV-denoised output image on the right, $\text{ISNR} = 6.1$ dB.

We use the freely available package `tvreg` from Pascal Getreuer available for download [36]. White Gaussian noise was added to produce an input $\text{SNR} = 10$ dB on the noisy input image. Three values of λ were tried, 20, 25, and 30, with 25 producing the best MSE result, which corresponded to an output $\text{SNR} = 16.1$ dB. Thus the SNR improvement $\text{ISNR} = 6.1$ dB. Figure 8.8-3 shows both the noisy input image on the left and the `tvdenoise` output on the right.

We see considerable noise reduction without the oversmoothing that generally occurs with the L^2 norm. On the other hand, the ISNR is not the best. While the algorithm runs very fast, even in MATLAB, iteration is required to find the best λ value. ■

8.9 IMAGE SUPERRESOLUTION

When additional versions of an image are available, we have an image sequence. If these frames are spatially aliased, then they may provide new unknown samples of the original image. *Superresolution* is a technique that combines (fuses) these aliased images to get a high-resolution (HR) output image from the input low-resolution (LR) frames. One can think of this as a type of restoration beyond the sampling limit of the individual frames, that is only permitted when multiple samplings of the data are present. So some slight camera and/or object motion between frames is crucial to the success of this method.

The first such result is due to Tsai and Huang [37] using a frequency-domain approach via fast Fourier transforms and was restricted to global translational motion between the frames. Many papers have appeared on superresolution since this first one. This first paper did not use the word superresolution (SR), but rather referred to image restoration and registration, which are constituent steps of SR.¹⁰

In matrix terms, the SR problem can be formulated, following Elad and Hel-Or [39], as

$$Y(n) = \mathcal{D}_n \mathcal{H}_n \mathcal{F}_n X + V(n), \quad n = 1, \dots, N,$$

where N is the number of images of X available and the $V(n)$ is a white Gaussian noise, independent from frame to frame. The 4-D matrices $\mathcal{D}_n \mathcal{H}_n \mathcal{F}_n$ reflect the deterministic distortion between images and are respectively decimation, blurring, and shift, or more generally warp. Taking the case where the decimation and blur are constant (i.e., $\mathcal{D}_n = \mathcal{D}$ and $\mathcal{H}_n = \mathcal{H}$ for all n), the maximum-likelihood¹¹ estimate can then be written as

$$\hat{X} = \arg \min_X \left\{ \sum_{n=1}^N [Y(n) - \mathcal{D} \mathcal{H} \mathcal{F}_n X]^t [Y(n) - \mathcal{D} \mathcal{H} \mathcal{F}_n X] / \sigma_V^2 \right\}, \quad (8.9-1)$$

where σ^2 is the variance of the added noise, assumed constant over the image sequence $Y(n)$. In practice, the shift operator \mathcal{F}_n has to be estimated from the LR image frames. Such is properly the topic of *motion estimation* to be covered in Chapter 11 on video processing. For now we just assume that this shift operator is known.

Taking a matrix derivative and setting to zero, we obtain

$$\mathcal{R} \hat{X} = P, \quad (8.9-2)$$

where $\mathcal{R} \triangleq \sum_{n=1}^N \mathcal{F}_n^t \mathcal{H}^t \mathcal{D}^t \mathcal{D} \mathcal{H} \mathcal{F}_n / \sigma^2$ and $P \triangleq \sum_{n=1}^N \mathcal{F}_n^t \mathcal{H}^t \mathcal{D}^t Y(n) / \sigma^2$. Noticing that we can cancel out the σ^2 term, and applying a steepest descent algorithm to find

¹⁰There is a new type of SR that tries to work within a single image to find missing HR data. It does this by a combination of template matching and scale-space (or SWT) analysis [38]. We do not discuss this method here.

¹¹Maximum likelihood estimates [1] treat the quantity to be estimated X as unknown but not random. They then try to maximize the pdf $f(\text{data}|X)$ over all possible values of the unknown quantity X . In estimation theory, this density is called the *a posteriori* density.

the solution, we obtain the iteration

$$\begin{aligned}
 \hat{\mathbf{X}}_{l+1} &= \hat{\mathbf{X}}_l + \mu [\mathbf{P} - \mathcal{R}\hat{\mathbf{X}}_l], \quad l = 1, 2, \dots \\
 &= \hat{\mathbf{X}}_l + \mu \sum_{n=1}^N \mathcal{F}_n^t \mathcal{H}^t \mathcal{D}^t [Y(n) - \mathcal{D}\mathcal{H}\mathcal{F}_n \hat{\mathbf{X}}_l] \\
 &= \hat{\mathbf{X}}_l + \mu \mathcal{H}^t \sum_{n=1}^N \mathcal{F}_n^t \mathcal{D}^t [Y(n) - \mathcal{D}\mathcal{F}_n \mathcal{H}\hat{\mathbf{X}}_l], \quad (8.9-3)
 \end{aligned}$$

if we assume that \mathcal{F}_n^t and \mathcal{H}^t commute (i.e., $\mathcal{F}_n^t \mathcal{H}^t = \mathcal{H}^t \mathcal{F}_n^t$), which is possible if the 4-D matrices are circulant. In practice this is approximated by various techniques such as windowing the input image frames.

Now if we define $\hat{\mathbf{Z}} \triangleq \mathcal{H}\hat{\mathbf{X}}$ and multiply (8.9-3) by \mathcal{H} , we get

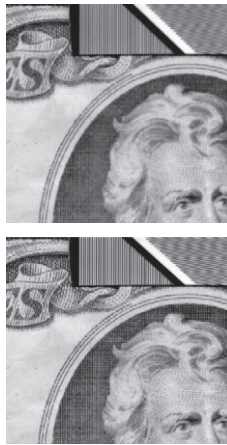
$$\hat{\mathbf{Z}}_{l+1} = \hat{\mathbf{Z}}_l + \mu \mathcal{H}\mathcal{H}^t \sum_{n=1}^N \mathcal{F}_n^t \mathcal{D}^t [Y(n) - \mathcal{D}\mathcal{F}_n \hat{\mathbf{Z}}_l], \quad l = 1, 2, \dots, \quad (8.9-4)$$

thus separating the image SR problem into two separable parts. First, perform iteration (8.9-4) to fuse the images into one at the high resolution (usually only a factor of two to four higher in each dimension), then perform a second step to allow image restoration on the noisy image $\hat{\mathbf{Z}}_\infty$ to get a final HR estimate $\hat{\mathbf{X}}$ based on the relation $\mathbf{Z} = \mathcal{H}\mathbf{X}$. In principle, this second step can be done by a conventional image restoration method, such as the ones discussed earlier in this chapter. For more details on this approach to SR, see [39].

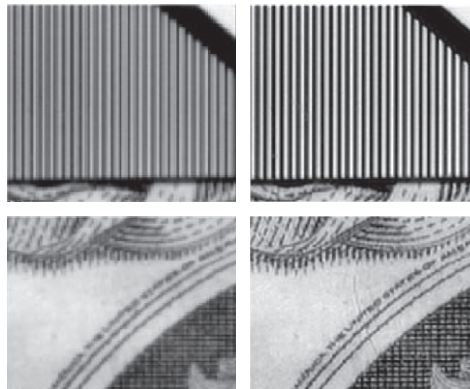
In practice, one may not know the exact values of the shifts or displacements between the input images. In this case, one has to use a motion estimation method to obtain these shifts with respect to one of them, a chosen *target image*. Basically, this will involve some kind of template matching, as discussed in Chapter 7. More detailed analysis of motion estimation, mainly for the video application, comes in Chapter 11.

Example 8.9-1: Image SR

This example is from Elad and Hel-Or [39] and consists of 16 images taken by a digital camera of size 300×360 pixels. The resolution was then increased by a factor of two in each dimension. The point spread function (psf) was assumed to be a Gaussian blur with operator-selected variance. Results are shown in Figure 8.9-1, with a close-up in Figure 8.9-2. Notice the increased sharpness in Figure 8.9-1, and notice the reduction in aliasing of the vertical lines in the close-up in Figure 8.9-2.

**FIGURE 8.9-1**

Results of image SR from Elad and Hel-Or [39, Figure 3]. (copyright IEEE, 2001)

**FIGURE 8.9-2**

Close-up of [Figure 8.9-1](#).

The success of SR depends on the accuracy of the registration between the LR frames. While aliasing is necessary to get the extra resolution (i.e., missing high-frequency content), the presence of aliasing makes the registration problem more difficult. See Vandewalle et al. [40] for an accurate method of measuring the global shift and rotation between aliased images.

8.10 COLOR IMAGE PROCESSING

Since color images are simply multicomponent or vector images, usually of the three components, red, green, and blue, one could apply all the methods of this chapter

to each component separately. Alternatively, one can develop vector versions of the estimation methods for application to the vector images. Usually, though, color image processing is conducted in the $Y' C_R C_B$ domain, by just processing the luma component Y' . For example, in color image deblurring, often just the luma image is restored, while the chroma components are left alone, owing to the lower chroma bandwidth of the HVS and lesser sensitivity to error. Regarding blur function estimation, often this is determined from the luminance data alone, and then the same blur function is assumed for the chrominance channels. Such relatively simple color processing, however, has been criticized as leading to *color shifts* that can sometimes leave unnatural colors in the restored images. Vector restoration procedures have been especially developed to deal with this problem. A special issue of *Signal Processing Magazine* [41] deals extensively with the overall topic of processing color images.

Of course, any color space can be used for image processing. Recently so-called *.raw data* has been made available as digital camera output. These data are direct from the sensors without any gamma correction. Also, in most cases they have not been demosaiced yet. In that case, there is the opportunity to perform restoration directly on the higher bitdepth sensor data.

CONCLUSIONS

This chapter introduced the problem of estimation and restoration for spatial systems. We presented both nonrecursive and recursive approaches for 2-D estimation. These methods extend the well-known 1-D discrete-time Wiener filter and predictor to the 2-D or spatial case. We considered recursive estimation, reviewed the 1-D Kalman filter, and then extended it to the 2-D case, where we saw that the state vector spread out to the width of the signal field and was therefore not as efficient as the 1-D Kalman filter for the chosen AR model. We then presented a reduced update version of 2-D recursive linear estimation that regains some of the lost properties. We also looked at space-variant extensions of these LSI models for adaptive Wiener and Kalman filtering and provided some examples. Then we considered estimation in the subband/wavelet domain as an extension of a simple adaptive method involving local mean and variance functions. We looked at so-called Bayesian methods using compound Gauss-Markov models and simulated annealing in their estimator solution. We introduced the problem of parameter estimation combined with image estimation and restoration and looked at how this EM technique can be carried to the SWT observations. We looked at some non-Bayesian approaches from the current literature. We considered the related problem of image superresolution, applicable to a situation where multiple aliased LR images of a scene are available and an HR result is desired. Finally, we briefly considered some issues in extending our restoration techniques to color images.

PROBLEMS

1. Carry out the indicated convolutions in (8.1–1) to obtain the covariance function of white noise of variance σ_w^2 having been filtered with the impulse response $g(n_1, n_2)$. Using the notation $g_-(n_1, n_2) \triangleq g(-n_1, -n_2)$, express your answer in terms of σ_w^2 and a direct convolution of g and g_-^* .
2. Extend the method in Example 8.1–3 to solve for the (1,1)-order NSHP predictor, where the coefficient support of the 2-D linear predictor is given as

$$\mathcal{R}_a = (0,0) \triangleq \{(1,0), (-1,1), (0,1), (1,1)\}.$$

Write the 4×4 matrix Normal equation for this case in terms of a homogeneous random field correlation function $R_x(m_1, m_2)$.

3. This problem concerns a least-squares version of 2-D linear prediction. The difference here is that we consider the signal $x(n_1, n_2)$ as nonrandom. Then instead of minimizing the mean-square error, we minimize the deterministic *least-squares error*

$$\mathcal{E} \triangleq \sum_{n_1=1}^{N_1-1} \sum_{n_2=1}^{N_2-1} (x(n_1, n_2) - \hat{x}(n_1, n_2))^2,$$

where $\text{supp}\{x\} = [0, N_1 - 1] \times [0, N_2 - 1]$, and the 2-D linear prediction \hat{x} is given as

$$\hat{x}(n_1, n_2) \triangleq a_{1,0}x(n_1 - 1, n_2) + a_{0,1}x(n_1, n_2 - 1).$$

Notice the limits of summation in \mathcal{E} that ensure we only reference existing data (i.e., nonzero values of the signal x).

- (a) Find the 2×2 matrix-vector equation that the coefficients must satisfy. Define the column vector $\mathbf{a} = [a_{1,0}, a_{0,1}]^T$.
 - (b) Solve the resulting equation for the predictor coefficients $a_{1,0}$ and $a_{0,1}$.
4. Write the equations that specify a 5×5 -tap FIR Wiener filter h with support on $[-2, +2]^2$ for the signal correlation function $R_x(m_1, m_2) = \sigma_x^2 \rho^{|m_1| + |m_2|}$, where σ_x^2 is a given positive value and ρ satisfies $|\rho| < 1$. Assume the observation equation is specified as

$$y(n_1, n_2) = 5x(n_1, n_2) + 3w(n_1, n_2),$$

where w is a white noise of variance σ_w^2 and $w \perp x$.

5. Reconsider problem 4, but now let the Wiener filter h have infinite support and actually write the solution in the frequency domain—i.e., find $H(\omega_1, \omega_2)$.
6. To determine a general causal Wiener filter, we need to first find the spectral factors of the noisy observations $S_y(\omega_1, \omega_2)$, in the sense of Theorem 8.2–1. Show that the power spectral density

$$S_y(\omega_1, \omega_2) = \frac{1}{1.65 + 1.6 \cos \omega_1 + 0.2 \cos \omega_2 + 0.16 \cos(\omega_1 - \omega_2)}$$

has stable and QP causal spectral factor

$$B_{++}(z_1, z_2) = \frac{1}{1 + 0.8z_1^{-1} + 0.1z_2^{-1}}.$$

What is $B_{--}(z_1, z_2)$ here? What is σ^2 ?

7. The 2-D Wiener filter has been designed using the orthogonality principle of linear estimation theory. Upon setting

$$\hat{x}(n_1, n_2) \triangleq \sum_{(k_1, k_2) \in \mathcal{R}_h} h_{k_1, k_2} y(n_1 - k_1, n_2 - k_2),$$

we found in the homogeneous, zero-mean case, that h is determined as the solution to the Normal equations

$$\sum_{(l_1, l_2) \in \mathcal{R}_h} h_{l_1, l_2} R_{yy}(k_1 - l_1, k_2 - l_2) = R_{xy}(k_1, k_2) \text{ for } (k_1, k_2) \in \mathcal{R}_h.$$

By ordering the elements $(k_1, k_2) \in \mathcal{R}_h$ onto vectors, this equation can be put into matrix-vector form and then solved for vector \mathbf{h} in terms of correlation matrix \mathbf{R}_{yy} and cross-correlation vector \mathbf{r}_{xy} , as determined by the chosen element order. In this problem, our interest is in determining the resultant mean-square value $E[|e(n_1, n_2)|^2]$ of the estimation error $e(n_1, n_2) = \hat{x}(n_1, n_2) - x(n_1, n_2)$.

(a) First, using the orthogonality principle, show that

$$\begin{aligned} E[|e(n_1, n_2)|^2] &= -E[x(n_1, n_2)e^*(n_1, n_2)] \\ &= E[|x(n_1, n_2)|^2] - E[x(n_1, n_2)\hat{x}^*(n_1, n_2)] \\ &= R_{xx}(0, 0) - \sum_{(k_1, k_2) \in \mathcal{R}_h} h_{k_1, k_2}^* R_{xy}(k_1, k_2). \end{aligned}$$

- (b) Then show that the solution to the Normal equations and the result of part (a) can be combined and written in matrix-vector form as

$$\sigma_e^2 = \sigma_x^2 - \mathbf{r}_{xy}^{*T} \mathbf{R}_{yy}^{-1} \mathbf{r}_{xy}.$$

Note that $E[|e(n_1, n_2)|^2] = \sigma_e^2$ since we assume the means of x and y are zero.

8. In the 2-D Kalman filter, what is the motivation for omitting updates that are far from the observations? How does the RUKF differ from the approximate RUKF? For constant coefficient AR signal models and observation models, we experimentally find that a steady state is reached fairly rapidly as we process into the image. What role does model stability play in this?

9. Write out the steady-state RUKF equations for a 1×1 -order $\oplus+$ model and update region

$$\begin{aligned} \mathcal{U}_{\oplus+}(n_1, n_2) = \{ & (n_1, n_2), (n_1 - 1, n_2), (n_1 - 2, n_2), \\ & (n_1 + 2, n_2 - 1), (n_1 + 1, n_2 - 1), (n_1, n_2 - 1), (n_1 - 1, n_2 - 1), \\ & (n_1 - 2, n_2 - 1) \}. \end{aligned}$$

Write these equations in terms of model coefficients $\{c_{10}, c_{01}, c_{11}, c_{-1,1}\}$ and assumed steady-state gain values

$$\{g(0,0), g(1,0), g(2,0), g(-2,1), g(-1,1), g(0,1), g(1,1), g(2,1)\}.$$

10. With reference to problem 9, list the various estimates provided by the steady-state RUKF in terms of their increasing 2-D delay. For example, the first estimate is the prediction estimate $\hat{x}_b^{(n_1, n_2)}(n_1, n_2)$ with a delay of $(-1, 0)$ (i.e., an advance of one pixel horizontally).
11. Continuing along the lines of problem 1 of Chapter 7, show how to efficiently calculate the local variance used in Section 8.4 by extending the box filter concept. Assume the box is $2M + 1 \times 2M + 1$. How many multiplies per pixel? Adds per pixel? How much intermediate storage is needed?
12. Show that the Markov random field satisfying the noncausal 2-D difference equation (8.6-1), with random input satisfying (8.6-2), has the PSD (8.6-3).
13. Derive the scalar (8.7-9) and (8.7-10) from the 4-D matrix equations (8.7-7) and (8.7-8). Make use of the fact that, in the homogeneous case,

$$\left(\hat{\mathcal{K}}_{XX}^{(k)} \right)_{n_1, n_2; n_1 - k_1, n_2 - k_2} = \left(\hat{\mathcal{K}}_{XX}^{(k)} \right)_{0,0; -k_1, -k_2},$$

for all (n_1, n_2) .

14. Using moment-generating functions and characteristic functions, show that the scalar Laplace random X variable can be written as the product of the square root of an exponential random variable Z and an independent Gaussian random variable U . Write the corresponding equation for probability densities.
15. Download the `tvreg` package [36] and repeat Example 8.8-3 for the range of $\lambda : 16-30$ at intervals of two, and plot your results in terms of ISNR. Which estimated image looks best to you?
16. Show that (8.9-2) results from taking the 2-D vector derivative¹² of the function in the argmax in (8.9-1) with regard to the 2-D vector X and setting it to the zero 2-D vector zero.

¹²This derivative is simply the 2-D vector composed of the partial derivatives of the function within the argmax in (8.9-1) with regard to each of the variables $X(n_1, n_2)$.

APPENDIX: RANDOM PROCESSES

Summary of Random Sequences and Processes

A random process is an extension of the concept of random variable as studied in a course on probability [1]. Basically, and generalizing the concept of random variable, the theory of random processes treats random 1-D functions. Usually they are random functions of time, but equally well this parameter could be space, such as along the scan line of an image or the acoustic signal from a line of sensors in geophysical exploration. In the discrete-time (discrete-parameter) case, we call this set of functions a random sequence. We can equally well think of this random sequence as an infinite set of random variables together with a time index n .

We start this section with a short review of basic probability. Then we summarize some important concepts for random sequences and random processes, whose knowledge is needed for both this and later chapters of this book.

Probability Review

A probability space is composed of a set of *outcomes* ζ of an experiment. The set of all outcomes is called the *sample space* Ω . Subsets of the space Ω are called *events*. A *field of events* \mathcal{F} is then defined to include all the important subsets of this sample space of outcomes Ω . Finally, a *probability measure* $P[A]$ is defined for each event A in the field \mathcal{F} . The resulting *probability space* is denoted (Ω, \mathcal{F}, P) . Every field of events must contain the *null event* ϕ and the *sure event* Ω . Also, for mathematical completeness, it must contain all countable intersections and unions of its sets (events).

The probability measure P must satisfy three axioms:

1. $P[A] \geq 0$ for all events $A \in \mathcal{F}$.
2. $P[\Omega] = 1$.
3. $P[A \cup B] = P[A] + P[B]$, for all events $A \cap B = \phi$, called the *null event*.

Now a random variable X is simply a function defined on the sample space $X(\zeta)$ that satisfies certain regularity properties:

1. The set $\{\zeta | X(\zeta) \leq x\}$ must be in the field of events \mathcal{F} for every $-\infty < x < +\infty$.
2. $P[\{\zeta | X(\zeta) = -\infty\}] = P[\{\zeta | X(\zeta) = +\infty\}] = 0$.

This random variable maps the events to the real line or the complex plane, thereby giving a numerical value for each outcome of the experiment.

For each random variable X defined on a probability space (Ω, \mathcal{F}, P) , we can construct a *cumulative distribution function* (CDF)

$$F_X(x) \triangleq P[\{\zeta | X(\zeta) \leq x\}].$$

Thus the distribution function $F_X(x)$ is just the probability of the event $\{\zeta | X(\zeta) \leq x\}$. For notational simplicity, we generally write such events as just $\{X \leq x\}$, and their

corresponding probability as

$$F_X(x) = P[X \leq x].$$

A valid distribution function must then have the following properties:

1. $F_X(-\infty) = 0$ and $F_X(+\infty) = 1$.
2. For all $x_1 \leq x_2$, we have $F_X(x_1) \leq F_X(x_2)$; that is, the distribution function is *monotonic nondecreasing*.
3. For the event $\{x_1 < X \leq x_2\}$, we have the probability $P[\{x_1 < X \leq x_2\}] = P[x_1 < X \leq x_2] = F_X(x_2) - F_X(x_1)$.

If the distribution function is differentiable, then we can define a *probability density function* (pdf) as

$$f_X(x) \triangleq \frac{dF_X(x)}{dx},$$

with the following properties:

1. $f_X(x) \geq 0$.
2. $\int_{-\infty}^{+\infty} f_X(x) dx = 1$.
3. $F_X(x) = \int_{-\infty}^x f_X(\eta) d\eta$.
4. $F_X(x_2) - F_X(x_1) = \int_{x_1}^{x_2} f_X(\eta) d\eta$, for all $x_2 \geq x_1$.

In basic probability, we generally study one or a few random variables and also functions of these random variables. For two random variables X and Y , their joint distribution function is given as

$$F_{X,Y}(x,y) \triangleq P[\{\zeta | X(\zeta) \leq x\} \cap \{\zeta | Y(\zeta) \leq y\}],$$

or with short notation,

$$F_{X,Y}(x,y) = P[\{X \leq x, Y \leq y\}] = P[X \leq x, Y \leq y],$$

with the following properties:

1. $F_{X,Y}(-\infty, -\infty) = F_{X,Y}(-\infty, +\infty) = F_{X,Y}(+\infty, -\infty) = 0$ and $F_{X,Y}(+\infty, +\infty) = 1$.
2. For all $x_1 \leq x_2$ and $y_1 \leq y_2$, we have $F_{X,Y}(x_1, y_1) \leq F_{X,Y}(x_2, y_2)$.
3. For the event $\{x_1 < X \leq x_2, y_1 < Y \leq y_2\}$, we have the probability

$$\begin{aligned} P[\{x_1 < X \leq x_2, y_1 < Y \leq y_2\}] &= P[x_1 < X \leq x_2, y_1 < Y \leq y_2] \\ &= F_{X,Y}(x_2, y_2) - F_{X,Y}(x_1, y_2) - F_{X,Y}(x_2, y_1) + F_{X,Y}(x_1, y_1), \end{aligned}$$

which, of course, must always be nonnegative for any valid joint distribution function.

Note that condition 3 is more than simple monotonicity of the marginal distributions F_X and F_Y . It can be seen as a kind of 2-D monotonic nondecreasing condition.

Correspondingly, the joint pdf, when it exists, is given as

$$f_{X,Y}(x,y) \triangleq \frac{\partial F_{X,Y}(x,y)}{\partial x \partial y},$$

with the following properties:

1. $f_{X,Y}(x,y) \geq 0$.
2. $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_{X,Y}(x,y) dx dy = 1$.
3. $F_{X,Y}(x,y) = \int_{-\infty}^x \int_{-\infty}^y f_{X,Y}(\eta, \nu) d\eta d\nu$.
4. $P[x_1 < X \leq x_2, y_1 < Y \leq y_2] = \int_{x_1}^{x_2} \int_{y_1}^{y_2} f_{X,Y}(\eta, \nu) d\eta d\nu$, whenever both $x_2 \geq x_1$ and $y_2 \geq y_1$.

In general, two random variables will be dependent, meaning that the value taken on by one affects the probability distribution of the other. In fact, this is the general case. In exceptional cases, two random variables can be *statistically independent*, or simply *independent*.

Definition 8.A–1: Independent Random Variables

Two random variables X and Y are said to be independent if their joint distribution function factors or separates as

$$F_{X,Y}(x,y) = F_X(x)F_Y(y) \quad \text{for all } -\infty < x, y < +\infty.$$

Equivalently, for joint pdf's,

$$f_{X,Y}(x,y) = f_X(x)f_Y(y) \quad \text{for all } -\infty < x, y < +\infty.$$

We can go on to generalize these concepts to three and more random variables coming to the concept of the random vector. In basic probability, we also study infinite sequences of random variables, but only in the *jointly independent* case, under the topics of Central Limit Theorems and Laws of Large Numbers [1].

Random Sequences

As mentioned in the introduction to this appendix, a random sequence generalizes the concept of random variable. Instead of defining just one function on the sample space, we define a sequence of such functions.

Definition 8.A–2: Random Sequence

Given a probability space (Ω, \mathcal{F}, P) , a random sequence is defined as the mapping $X(n, \zeta)$ for each outcome $\zeta \in \Omega$, satisfying the property that for each fixed $-\infty < n < +\infty$, the mapping must be a random variable.

So we can look at a random sequence as an infinite sequence of possibly dependent random variables. For fixed time n , we have a random variable, and for fixed

outcome ζ , we have a deterministic sequence. If we sample a random sequence at a finite set of times, we have a random vector. Looking at a specific $K \geq 1$ times n_1, n_2, \dots, n_K , we can define the K th-order distribution function

$$F_X(x_1, x_2, \dots, x_K; n_1, n_2, \dots, n_K) \triangleq P[X(n_1) \leq x_1, X(n_2) \leq x_2, \dots, X(n_K) \leq x_K],$$

to probabilistically describe this K -dimensional sample of the random sequence. If we know these distribution functions for all $1 \leq K < \infty$, and for all times n_1, n_2, \dots, n_K , then we say that we have *characterized* or *defined* the random sequence $X(n)$. With some possibility of confusion, we often abbreviate this notation to simply

$$F_X(x_{n_1}, x_{n_2}, \dots, x_{n_K}) = F_X(x_{n_1}, x_{n_2}, \dots, x_{n_K}; n_1, n_2, \dots, n_K),$$

by leaving off explicit notation of the times involved in this K th-order distribution function. The corresponding result for density functions is

$$f_X(x_1, x_2, \dots, x_K; n_1, n_2, \dots, n_K) \triangleq \frac{\partial^K F_X(x_1, x_2, \dots, x_K; n_1, n_2, \dots, n_K)}{\partial x_1 \partial x_2 \dots \partial x_K},$$

with simplified notation

$$f_X(x_{n_1}, x_{n_2}, \dots, x_{n_K}) = f_X(x_{n_1}, x_{n_2}, \dots, x_{n_K}; n_1, n_2, \dots, n_K).$$

As a partial characterization of random process $X(n)$, we can define the useful first and second moment functions:

Mean function:

$$\mu_X(n) \triangleq E[X(n)] = \int_{-\infty}^{+\infty} x_n f_X(x_n) dx_n.$$

Correlation function:

$$R_X(n_1; n_2) \triangleq E[X(n_1)X^*(n_2)] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_{n_1} x_{n_2}^* f_X(x_{n_1}, x_{n_2}) dx_{n_1} dx_{n_2}.$$

Covariance function:

$$\begin{aligned} K_X(n_1; n_2) &\triangleq E[(X(n_1) - \mu_X(n_1))(X(n_2) - \mu_X(n_2))^*] \\ &= R_X(n_1; n_2) - \mu_X(n_1)\mu_X^*(n_2). \end{aligned}$$

To these functions we can add the *variance function*, $\sigma_X^2(n) \triangleq K_X(n; n) = E[|X(n)|^2]$.

Stationary Random Sequences

The distribution functions that characterize a random sequence may not change with time. In that case, the complete set of distribution functions will only be a function

of the intersample intervals involved. In such a case, the mean function $\mu_X(n)$ will be constant, and the correlation function $R_X(n_1; n_2)$ will be just a function of the time difference between the two observation times n_1 and n_2 .

Definition 8.A–3: Stationary Random Sequence

A random sequence $X(n)$ is stationary if for every $K \geq 1$, the K th-order distribution functions are invariant with respect to time shift m , for all $-\infty < m < +\infty$; that is,

$$F_X(x_1, x_2, \dots, x_K; n_1, n_2, \dots, n_K) = F_X(x_1, x_2, \dots, x_K; n_1 + m, n_2 + m, \dots, n_K + m),$$

and for all sample times $-\infty < n_1, n_2, \dots, n_K < +\infty$.

Note that here we must use a complete rather than the short notation $F_X(x_{n_1}, x_{n_2}, \dots, x_{n_K})$ in order to make this concept clear.

In terms of the first- and second-order moment functions, we have the weaker concept of wide-sense stationarity.

Definition 8.A–4: Wide-Sense Stationarity

A random sequence $X(n)$ is wide-sense stationary (WSS) if

1. $\mu_X(n) = \mu_X(0)$, a constant, and
2. $R_X(n_1; n_2) = R_X(n_1 - n_2; 0)$ for all n_1 and n_2 ,
3. $K_X(n_1; n_2) = K_X(n_1 - n_2; 0)$ for all n_1 and n_2 .

For a WSS random sequence, we can write the basic moment functions simply as $\mu_X \triangleq \mu_X(0)$ and $R_X(m) \triangleq R_X(m; 0)$, a one-parameter function of the variable m . Similarly, for the covariance function, we can write $K_X(m) \triangleq K_X(m; 0)$. Thus we have the following for any WSS random sequence $X(n)$:

$$\mu_X = E[X(n)],$$

$$R_X(m) = E[X(n+m)X^*(n)],$$

$$K_X(m) = E[(X(n+m) - \mu_X)(X(n) - \mu_X)^*],$$

independent of the value of n .

The Fourier transform of the WSS correlation function $R_X(m)$ is called the *power spectral density* (PSD),

$$S_X(\omega) \triangleq \sum_{m=-\infty}^{+\infty} R_X(m) \exp -jm\omega.$$

The PSD has the interpretation of average power density on the frequency axis $-\infty < \omega < +\infty$. As such, we can find the average power of $X(n)$ in a frequency band $[\omega_0, \omega_1]$ by integrating the PSD over that band [1].

Example 8.A–1: Geometric Correlation Function

Consider the correlation function

$$R_X(m) = \sigma_X^2 \rho^{|m|} + |\mu_X|^2,$$

where $\mu_X, \sigma_X (> 0)$ and ρ are given constants with $|\rho| < 1$. When $|m|$ is very large, we see that $R_X(m) \approx |\mu_X|^2$; that is, $X(n+m)$ and $X(n)$ are approximately uncorrelated. When $m=0$, we have $R_X(0) = \sigma_X^2 + |\mu_X|^2 = E[|X(n)|^2]$, the average power in the random sequence.

To find the corresponding PSD $S_X(\omega)$, we plug into the definition to get

$$\begin{aligned} S_X(\omega) &= \sum_{-\infty < m < +\infty} R_X(m) \exp -jm\omega \\ &= \sum_{-\infty < m < +\infty} (\sigma_X^2 \rho^{|m|} + |\mu_X|^2) \exp -jm\omega \\ &= \sigma_X^2 \sum_{-\infty < m < +\infty} \rho^{|m|} e^{-jm\omega} + |\mu_X|^2 \sum_{-\infty < m < +\infty} \exp -jm\omega \\ &= \sigma_X^2 \left[\sum_{m=-\infty}^{-1} (\rho e^{+j\omega})^{-m} + \sum_{m=0}^{+\infty} (\rho e^{-j\omega})^m \right] + 2\pi |\mu_X|^2 \delta(\omega) \\ &= \sigma_X^2 \left[\sum_{m'=1}^{+\infty} (\rho e^{+j\omega})^{m'} + \sum_{m=0}^{+\infty} (\rho e^{-j\omega})^m \right] + 2\pi |\mu_X|^2 \delta(\omega), \quad \text{with } m' \triangleq -m, \\ &= \sigma_X^2 \left(\frac{\rho e^{+j\omega}}{1 - \rho e^{+j\omega}} + \frac{1}{1 - \rho e^{-j\omega}} \right) + 2\pi |\mu_X|^2 \delta(\omega) \\ &= \sigma_X^2 \frac{(1 - \rho^2)}{(1 + \rho^2) - 2\rho \cos \omega} + 2\pi |\mu_X|^2 \delta(\omega) \quad \text{for } -\pi \leq \omega \leq +\pi. \end{aligned}$$

So the PSD has a Dirac impulse of area $2\pi |\mu_X|^2$ at the origin $\omega = 0$, and a continuous term with peak value σ_X^2 at the origin, smoothly decreasing in both directions, to a minimum value $\sigma_X^2 (1 - \rho^2) / (1 + \rho^2)$ at $\omega = \pm\pi$. ■

Random Processes

Random processes exist in continuous rather than discrete time. As such, for each outcome zeta ζ in the sample space Ω , we now have a continuous-time function.

Definition 8.A–5: Random Process

Given a probability space (Ω, \mathcal{F}, P) , a random process is defined as the mapping $X(t, \zeta)$ for each outcome $\zeta \in \Omega$, satisfying the property that for each fixed $-\infty < t < +\infty$, the mapping must be a random variable—i.e., the set $\{\zeta | X(t, \zeta) \leq x\}$ must be in the field of events \mathcal{F} . ■

Similarly to the case for a random sequence, we can define mean, correlation, and covariance functions for a random process. The concept of stationarity for random process is analogous to that of random sequences. Stationary random processes have power spectral densities that are the Fourier transforms of their stationary correlation functions. There are also sampling theorems that apply for bandlimited stationary random processes. Much more information on random sequences and processes can be found in textbooks such as [1].

REFERENCES

- [1] H. Stark and J. W. Woods, *Probability and Random Processes with Appl. to Signal Process.*, 3rd Ed., Prentice-Hall, EnglewoodCliffs, NJ, 2002.
- [2] J. R. Jain and A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Trans. Comm.*, vol. COM-29, pp. 1799–1804, December 1981.
- [3] B. Girod, "The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences," *IEEE J. Select. Areas in Comm.*, vol. SAC-5, pp. 1140–1154, August 1987.
- [4] J. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, EnglewoodCliffs, NJ, 1990.
- [5] M. P. Ekstrom and J. W. Woods, "Two-Dimensional Spectral Factorization with Applications in Recursive Digital Filtering," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 115–128, April 1976.
- [6] M. D. Robinson and D. G. Stork, "Joint Design of Lens Systems and Digital Image Processing," *Proc. SPIE Intl. Optical Design Conf.*, vol. 6342 (63421G), 2006.
- [7] J. W. Woods and V. K. Ingle, "Kalman Filtering in Two Dimensions: Further Results," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 29, pp. 188–197, 1981.
- [8] F.-C. Jeng and J. W. Woods, "Inhomogeneous Gaussian Image Models for Estimation and Restoration," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 36, pp. 1305–1312, August 1988.
- [9] M. R. Banham and A. K. Katsaggelos, "Digital Image Restoration," *IEEE Signal Process. Magazine*, pp. 24–41, March 1997. (See also "RUKF Performance Revisited" in *Signal Processing Forum* section of the November 1997 issue of the same magazine, p. 12 and 14.)
- [10] B. Friedland, "On the Properties of Reduced-Order Kalman Filters," *IEEE Trans. Auto. Control*, vol. 34, pp. 321–324, March 1989.
- [11] A. J. Patti, A. M. Tekalp, and M. I. Sezan, "A New Motion-Compensated Reduced-Order Model Kalman Filter for Space-Varying Restoration of Progressive and Interlaced Video," *IEEE Trans. Image Process.*, vol. 7, pp. 543–554, April 1998.
- [12] D. L. Angwin and H. Kaufman, "Image Restoration Using Reduced Order Models," *Signal Processing*, vol. 16, pp. 21–28, January 1988.
- [13] J. Kim and J. W. Woods, "A New Interpretation of ROMKF," *IEEE Trans. Image Process.*, vol. 6, pp. 599–601, April 1997.
- [14] R. Wallis, "An Approach to the Space Variant Restoration and Enhancement of Images," *Proc. Sympos. Current Math. Problems in Image Science*, Naval Postgraduate School, Monterey, CA, pp. 107–111, November 1976.
- [15] D. L. Donoho, "De-Noising by Soft-Thresholding," *IEEE Trans. Inform. Theory*, vol. 41, pp. 613–627, May 1995.

- [16] H. Zhang, A. Nosratinia, and R. O. Wells, Jr., "Image Denoising via Wavelet-Domain Spatially Adaptive FIR Wiener Filtering," *Proc. ICASSP 2000*, Vancouver, BC, pp. 2179–2182, May 2000.
- [17] S. G. Chang, B. Yu, and M. Vetterli, "Spatially Adaptive Wavelet Thresholding with Context Modeling for Image Denoising," *IEEE Trans. Image Process.*, vol. 9, pp. 1522–1531, September 2000.
- [18] J. W. Woods, "Two-dimensional Discrete Markovian Random Fields," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 232–240, March 1972.
- [19] J. Besag, "On the Statistical Analysis of Dirty Pictures," *J. Royal Statist. Soc. B*, vol. 48, pp. 259–302, 1986.
- [20] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-6, pp. 721–741, November 1984.
- [21] F.-C. Jeng and J. W. Woods, "Compound Gauss-Markov Random Fields for Image Estimation," *IEEE Trans. Signal Process.*, vol. 39, pp. 683–697, March 1991.
- [22] S. Rastogi and J. W. Woods, "Image Restoration by Parallel Simulated Annealing Using Compound Gauss-Markov Models," *Proc. ICASSP-91*, Toronto, Canada, 1991.
- [23] R. Molina, A. K. Katsaggelos, J. Mateos, A. Hermoso, and C. A. Segall, "Restoration of Severely Blurred High Range Images Using Stochastic and Deterministic Relaxation Algorithms in Compound Gauss-Markov Random Fields," *Pattern Recognition*, Elsevier Pergamon, vol. 33, pp. 555–571, 2000.
- [24] G. K. Chantas, N. P. Galatsanos, and A. C. Likas, "Bayesian Restoration Using a New Nonstationary Edge-Preserving Image Prior," *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 2987–2997, October 2006.
- [25] R. L. Lagendijk, J. Biemond, and D. E. Boekee, "Identification and Restoration of Noisy Blurred Images Using the Expectation-Maximization Algorithm," *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 38, pp. 1180–1191, July 1990.
- [26] A. P. Demster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.
- [27] J. Kim and J. W. Woods, "Image Identification and Restoration in the Subband Domain," *IEEE Trans. Image Process.*, vol. 3, pp. 312–314, 1994 plus "Erratum," pp. 873, 1994.
- [28] W. T. Freeman and E. H. Adelson, "The Design and Use of Steerable Filters," *IEEE Pattern Anal. Machine Intell.* (PAMI), vol. 13, no. 9, pp. 891–906, 1991.
- [29] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1328–1351, November 2003.
- [30] S. Kotz, T. J. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations*, Birkhauser, Boston, MA, 2001.
- [31] S. Lyu and E. P. Simoncelli, "Modeling Multiscale Subbands of Photographic Images with Fields of Gaussian Scale Mixtures," *IEEE Trans. Pattern Anal. and Machine Intell.* (PAMI), vol. 31, no. 4, pp. 693–706, April 2009.
- [32] A. Buades, B. Coll, and J.-M. Morel, "A Non-local Algorithm for Image Denoising," *Proc. IEEE Computer Vision and Pattern Recog.* (CVPR), 2005.
- [33] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, August 2007.

- [34] B. R. Hunt, "The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer," *IEEE Trans. Computers*, vol. C-22, no. 9, pp. 805–812, September 1973.
- [35] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and Robust Multiframe Super Resolution," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1327–1344, October 2004.
- [36] P. Getreuer, *tvreg: Variational Imaging Methods for Denoising, Deconvolution, Inpainting, and Segmentation*. Available at <http://www.sciweavers.org/sourcecode/tvreg-variational-imaging-methods-denoising-deconvolution-inpainting-and-segmentation>.
- [37] R. Y. Tsai and T. S. Huang, "Moving Image Restoration and Registration," *IEEE ICASSP*, pp. 418–421, April 1980.
- [38] D. Glasner, S. Bagon, and M. Irani, "Super-Resolution from a Single Image," International Conf. on Computer Vision (ICCV), 2009.
- [39] M. Elad and Y. Hel-Or, "A Fast Super-Resolution Reconstruction Algorithm for Pure Translational Motion and Common Space-Invariant Blur," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1187–1193, August 2001.
- [40] P. Vandewalle, S. Susstrunk, and M. Vetterli, "A Frequency Domain Approach to Registration of Aliased Images with Application to Super Resolution," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 1–14.
- [41] H. J. Trussell, E. Saber, and M. Vrhel, Eds., "Color Image Processing Special Issue," *IEEE Signal Process. Magazine*, vol. 22, January 2005.

Digital Image Compression

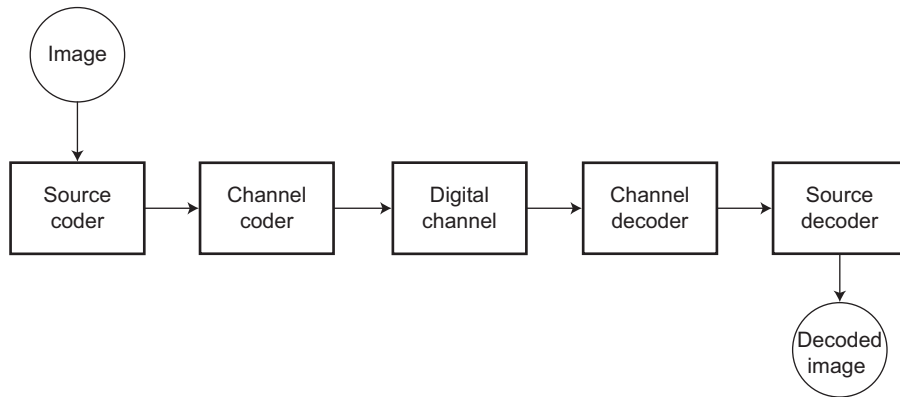
9

Images are perhaps the most influential of media that we encounter on a daily basis. This chapter will apply 2-D statistical signal processing to their compression for both efficient transmission and storage. We consider a generic model consisting of first transformation, then quantization, and then entropy coding. We cover the popular DCT-based image coding as well as SWT-based scalable image coders such as JPEG 2000. Finally, we present some current ideas for improving image compression.

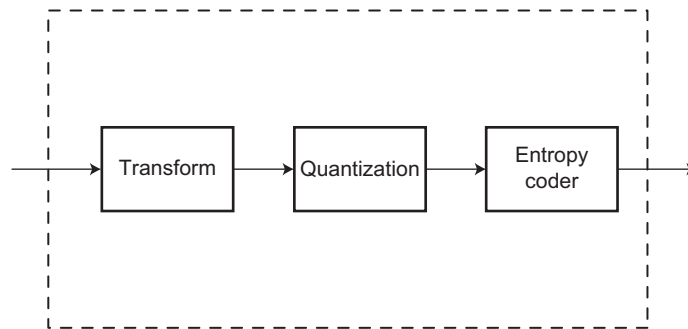
9.1 INTRODUCTION

With reference to [Figure 9.1–1](#), we see a universal diagram for the transmission or storage of digital images. On the extreme left, we have the input image file. We consider the case where the input image is digital, with word format either floating point or fixed point. The theory, though, assumes that it is composed of real numbers, and, in fact, our statistical model for the input image is a continuous-valued random field. From the viewpoint of the designer of an image communication system, the image is certainly unknown. Over the lifetime of the communications device, we see an ensemble of images that will be transmitted over it. This ensemble will have various statistical properties, such as histogram, joint histogram, mean function, correlation function, etc. So, it is reasonable to impose a statistical model for the *image source*.

The output of this image source is input to the *source coder*, whose purpose is to compress the image file by representing it as a finite string of binary digits (bits). If the input image is finite wordlength, then the source coder can be *lossless*, meaning that the input image can be reconstructed exactly from the compressed data. Similarly, a *lossy* coder is one for which only approximate reconstruction of the input image is possible. Lossy source coding is the main topic of this chapter. Next in the general communication diagram of [Figure 9.1–1](#) is the *channel coder*, whose purpose is to adapt or strengthen the compressed bitstream to survive the digital channel. One way it does this is by appending error correction bits to the compressed data words that make up the source-coded bitstream. Examples of digital channels are digital radio and TV broadcasts, optical fiber lines, cable modems, cell phone and Wi-Fi networks, and digital storage channels as found in disk memory systems. At a certain level of abstraction, we have the Internet and its protocol as a very common

**FIGURE 9.1-1**

Generic digital image communication system.

**FIGURE 9.1-2**

Generic source-coding system diagram.

collection of digital channels. The remaining parts of Figure 9.1-1 show the decoding. First the *channel decoder* attempts to recover the source coded bitstream, then the *source decoder* tries to reconstruct the source image. Figure 9.1-2 shows a useful decomposition valid for many source coders, which consist of a *transformation*, followed by a *quantizer* and then an *entropy coder*.

The purpose of the transformation is to remove or suppress the redundant parts of the data, hopefully yielding components that are independent or at least uncorrelated. Examples are differential *PCM* (DPCM)¹, orthonormal transforms based on frequency decompositions (DFT, DCT, and some SWTs), and the optimal Karhunen-Loeve transform (KLT) [1]. These transforms can be applied to the entire image, or

¹In differential pulse-code modulation the transform and the quantizer are linked together in a feedback loop. Therefore it is not right to say that the transform precedes the quantizer always. In DPCM the roles of the transform and the quantizer are intertwined.

as is commonly done, to relatively small blocks of pixel values. Usually these blocks do not overlap, but in some methods (e.g., *lapped orthogonal transform* [LOT] and SWT) they do. The quantization block in Figure 9.1–2 is very important because this is where the actual data compression occurs by a shortening of the data wordlength. The transform block is invertible, so that no data compression actually occurs there. A quantizer is a nonlinear, zero-memory device that chooses representative values for ranges of input data coming from the transform, one at a time, called *scalar quantization*, or several at a time, called *vector quantization*. Most quantizers are scalar, and common examples are uniform quantization (also called an A/D converter), nonuniform quantization, optimal uniform, optimal nonuniform, etc. The entropy coder block in Figure 9.1–2 converts the representative values, outputs of the quantizer, to efficient variable-length codewords. If we omit this last block, then the quantizer output is converted into fixed-length codewords, which are usually less efficient. Examples of variable-length codes are Huffman and arithmetic codes, which will be briefly described in Section 9.4 (see also Section 9.10 Appendix on Information Theory).

9.2 TRANSFORMATION

The role of the transformation in data compression is to decorrelate, and more generally, remove dependency between the data values, but without losing any information. For example, the transformation should be invertible, at least with infinite wordlength arithmetic. For Gaussian random fields, rate-distortion theory [1, 2] states that the optimal transformation for this purpose is the KLT. The transform can be applied in small blocks or to the entire image. Common block transforms include the DFT, DCT, and the overlapped extension of DCT called LOT. The DCT is motivated by being close to the KLT for a 1-D first-order Gauss-Markov random sequence, when the correlation coefficient $\rho \approx 1.0$. The practical advantage of the DCT in these cases is that it has a fast algorithm, unlike the general KLT. For images, the 2-D DCT is motivated by the separable Gaussian random field with the correlation function given as a separable product of the 1-D Markov correlation function,

$$R_x(m_1, m_2) = \sigma_x^2 \rho_1^{|m_1|} \rho_2^{|m_2|}, \quad (9.2-1)$$

where ρ_1 and ρ_2 are the horizontal and the vertical correlation coefficients, respectively, $0 < \rho_1, \rho_2 < 1$, which should both be close to one.

DCT

As we have seen in Chapter 4, for a data array of rectangular support $[0, N_1 - 1] \times [0, N_2 - 1]$, the 2-D DCT is given as

$$X_C(k_1, k_2) \triangleq \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} 4x(n_1, n_2) \cos \frac{\pi k_1}{2N_1} (2n_1 + 1) \cos \frac{\pi k_2}{2N_2} (2n_2 + 1),$$

for $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. Otherwise, $X_C(k_1, k_2) \triangleq 0$. However, in this chapter, we only consider $(k_1, k_2) \in [0, N_1 - 1] \times [0, N_2 - 1]$. For the common block-DCT and an input image of size $L_1 \times L_2$, we decompose it into $N_1 \times N_2$ blocks and perform the DCT for each successive block. The total number of blocks then becomes $\lceil L_1/N_1 \rceil \times \lceil L_2/N_2 \rceil$, where the last block in each line and column may be a partial block. This transforms the image data into a sequence of DCT coefficients, which can be stored blockwise or can be stored with all like-frequency coefficients together, effectively making a small ‘image’ out of each DCT coefficient. Of these, the so-called DC image made up of the $X_C(0, 0)$ coefficients is a thumbnail version of the original image, and, in fact, is just an $(N_1 \times N_2) \downarrow$ subsampled version of the original after $N_1 \times N_2$ box filtering.

Owing to the nearness of a typical natural image correlation function to (9.2-1), the DCT coefficients in a given block are approximately uncorrelated. We can expect the coefficients from different blocks to be approximately uncorrelated because of their distance from one another, if the blocksize is not too small. Even for neighboring blocks, the argument of uncorrelatedness is quite valid for the upper frequency or AC coefficients and is only significantly violated by the DC coefficients and a few low-frequency coefficients. This interblock coefficient-correlation effect is almost ignored in standard coding algorithms such as JPEG, where only the DC coefficients of blocks are singled out for further decorrelation processing, usually consisting of a prediction followed by a coding of the prediction residual.

A key property of a transform like DCT used in data compression is that it is *orthonormal*, with its transformation matrix being called *unitary*. This means that there is a Parseval-like relation between the sum of the magnitude-square values in the transform space (i.e., the coefficients) and the data or image space. Since the transform is linear and constant, this energy balance must also hold for the coding error. So if we approximate these coefficients via quantization, and are concerned with minimizing the sum magnitude-squared error, we can just as well do this in the coefficient or transform domain, where each coefficient can be quantized (or coded more generally) *separately*, using a *scalar quantizer*,

$$\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} (x(n_1, n_2) - \hat{x}(n_1, n_2))^2 = \frac{1}{4N_1N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w(k_1)w(k_2) \times \left(X_C(k_1, k_2) - \hat{X}_C(k_1, k_2) \right)^2.$$

A generalization of the block-DCT transformation is the fast LOT [3]. Here, overlapping 16×16 LOT transform blocks are made up by cascading neighboring 8×8 DCT blocks with an orthogonalizing transformation on their outputs. Actually, this is done in one dimension and then the procedure is just copied to the 2-D case, via row-column separable processing. An advantage of the LOT is that the objectionable blocking effect of DCT compression is reduced by the overlapping LOT input windows. Also, the LOT respects the digital signal processing theoretical method of

2-D sectioned convolution (cf. Section 4.6) by employing block overlap in implementation of its block processing. Another way of providing overlapped blocks, and perhaps eliminating blocks altogether, is the SWT.

SWT

Subband/wavelet transformation (SWT) can be viewed as a generalization of block-based DCT. This is because there is a filter bank interpretation of the block-based DCT, wherein the filter impulse responses are the DCT basis functions, and the decimation is $(N_1 \times N_2) \downarrow$ downsampling. Usually in SWTs, the filter support is larger than the decimation ratio, which is normally $(2 \times 2) \downarrow$. So, SWT is seen as a generalization of DCT to overlapping blocks, with expected reductions or elimination of blocking artifacts. A general subband analysis/synthesis bank is shown in Figure 9.2–1.

Usually more stages of subband decomposition are conducted than just this one level. In the common dyadic (a.k.a. octave and wavelet) decomposition, each resulting LL subband is further split recursively, as illustrated in Figure 9.2–2, showing a three-level subband/wavelet decomposition.

While this dyadic decomposition is almost universal, it is known to be suboptimal when a lot of high spatial frequency information is present (e.g., the well-known test image *Barbara*). When the subband splitting is optimized for a particular set of image statistics, the name *wavelet packet* [4] has emerged for the corresponding transformation.

In summary, SWT is seen to be a generalization of the block-DCT- and LOT-based transformations and, as such, can only offer the possibility for better performance.

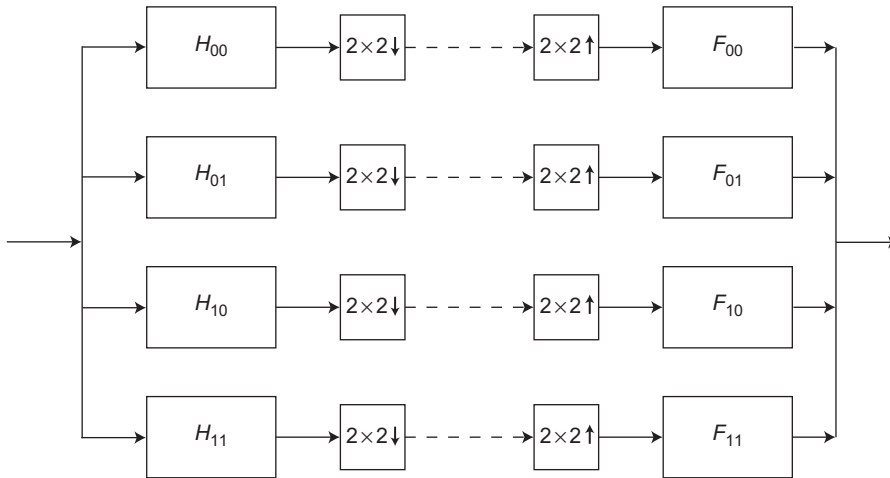


FIGURE 9.2–1

A general 2-D SWT or inverse SWT analysis/synthesis bank.

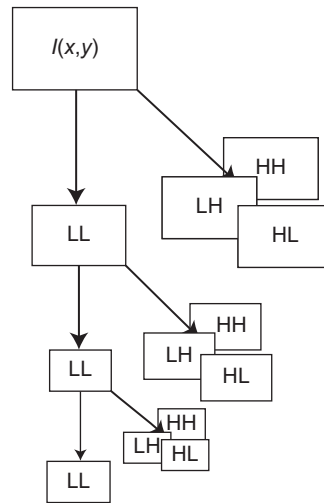
**FIGURE 9.2-2**

Illustration of dyadic (octave, wavelet) subband decomposition to three levels.

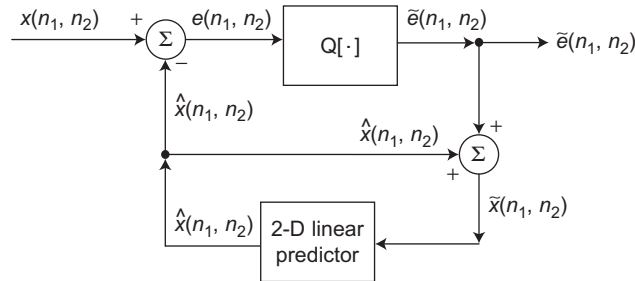
**FIGURE 9.2-3**

Diagram of a 2-D DPCM coder.

Best performance in this transform class can then be obtained by optimizing over the subband/wavelet filters used and the nature and manner of the subband splittings. The goal of all such transformations is to decorrelate (i.e., remove linear dependencies in) the data prior to quantization.

DPCM

A 2-D extension of *differential pulse-code modulation* (DPCM) has been extensively used for image compression. We see in Figure 9.2-3 that the main signal flow path across the top of the figure quantizes an error signal that is generated via a prediction $\hat{x}(n_1, n_2)$ that is fed back and, in turn, generated from only the *past* values of

the quantized error signal $\tilde{e}(n_1, n_2)$. Here, the transformation and the quantization are intertwined, not matching our paradigm of transform coming before quantizer (see Figure 9.1–2). We will look at quantizers in some detail in the next section, but for now, just consider it a device that approximates the incoming prediction error $e(n_1, n_2)$ with a finite number of levels. The signal estimate $\hat{x}(n_1, n_2)$ is produced by 2-D linear (1,0)-step prediction based on its input $\tilde{x}(n_1, n_2)$. Note that this feedback loop can also be run at the decoder, and, in fact, it *is* the decoder, with $\tilde{x}(n_1, n_2)$ as the decoder output. Here, we assume that the quantized error signal $\tilde{e}(n_1, n_2)$ is somehow losslessly conveyed to the receiver and its source decoder; that is, no channel errors occur.

One nice property of 2-D DPCM that directly extends from the 1-D case can be seen now. Consider the overall error at any given data location (n_1, n_2) :

$$\begin{aligned} x(n_1, n_2) - \tilde{x}(n_1, n_2) &= [x(n_1, n_2) - \hat{x}(n_1, n_2)] - [\tilde{x}(n_1, n_2) - \hat{x}(n_1, n_2)] \\ &= e(n_1, n_2) - \tilde{e}(n_1, n_2), \end{aligned}$$

as can be seen from the two equations at the summing junctions in Figure 9.2–3. So if we design the quantizer Q to minimize the MSE between its input and output, we are also finding the quantizer that minimizes the MSE at the DPCM decoder output. This very useful property would be lost if we moved the quantizer Q to the right and outside of the feedback loop. This is because a prediction-error filter does not constitute an orthogonal transformation, and quantization errors would accumulate at the decoder output.

There remains the problem of how to calculate the 2-D linear predictor coefficients. If we are concerned with high-quality (read high-bitrate) coding, then we can expect that $\tilde{x}(n_1, n_2) \approx x(n_1, n_2)$ to a sufficient degree of approximation to permit use of the linear (1,0)-step prediction filter that can be designed based on the quantization-noise-free input $x(n_1, n_2)$ itself. At low bitrates, some optimization can be done iteratively, based on starting with this filter, then generating the corresponding $\tilde{x}(n_1, n_2)$, and then generating a new linear (1,0)-step prediction filter, and so on.² However, DPCM performance deteriorates at low bitrates.

9.3 QUANTIZATION

The quantization operation is one of truncation or rounding. Figure 9.3–1 shows a quantizer Q , with input variable x and output variable $\hat{x} = Q(x)$, and Figure 9.3–2 shows its quantizer function $Q(x)$ plotted versus input x , which may be continuous valued or already quantized on a relatively fine scale to that of this quantizer.

²Starting this second time through, the filter is really a prediction estimator since its *design input* is \tilde{x} and not x anymore.

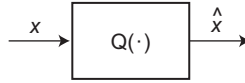


FIGURE 9.3–1

A scalar quantizer.

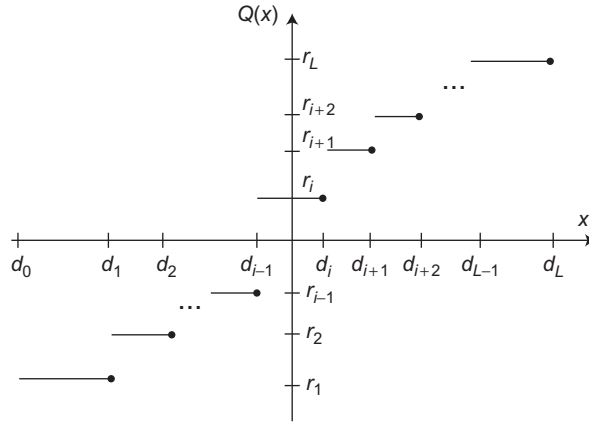


FIGURE 9.3–2

A quantizer characteristic that “rounds to the left.”

The *scalar quantizer* (SQ) is specified in general by its number of output values or levels L , its *decision levels* d_i , and its *representation levels* r_i , also the output values. The defining equation for the quantizer function is then

$$Q(x) \triangleq \begin{cases} r_1, & d_0 < x \leq d_1, \\ \vdots & \vdots \\ r_i, & d_{i-1} < x \leq d_i, \\ \vdots & \vdots \\ r_L, & d_{L-1} < x \leq d_L. \end{cases}$$

Here, we have taken the input range cells or *bins* as $(d_{i-1}, d_i]$, but they could equally be $[d_{i-1}, d_i)$. With reference to Figure 9.3–2, the total range of the quantizer is $[r_1, r_L]$ and its total domain is $[d_0, d_L]$, where normally $d_0 = x_{\min}$ and $d_L = x_{\max}$. For example, in the Gaussian case, we would have $d_0 = x_{\min} = -\infty$, and $d_L = x_{\max} = +\infty$.

We define the quantizer error $e_Q(x) \triangleq Q(x) - x = \hat{x} - x$, and measure the distortion as $d(x, \hat{x}) \triangleq |e_Q|$ or more generally $|e_Q|^p$, for positive integers p .

The average quantizer distortion D is given as

$$\begin{aligned}
 D &= E[d^2(x, \hat{x})] \\
 &= \int_{-\infty}^{+\infty} d^2(x, \hat{x}) f_x(x) dx \\
 &= \sum_{i=1}^L \int_{d_{i-1}}^{d_i} |x - r_i|^2 f_x(x) dx,
 \end{aligned}$$

where f_x is the pdf of the assumed random variable x .

Uniform Quantization

For $1 \leq i \leq L$, set $d_i - d_{i-1} = \Delta$, called the uniform quantizer *step size*. To complete the uniform quantizer, we set the representation value as

$$r_i \triangleq \frac{1}{2}(d_i + d_{i-1}), \quad (9.3-1)$$

at the center of the input bin $(d_{i-1}, d_i]$, for $1 \leq i \leq L$.

Example 9.3-1: Nine-Level Uniform Quantizer on $|x| \leq 1$

Let $x_{\min} = -1$ and $x_{\max} = +1$. Then we set $d_0 = -1$ and $d_L = +1$ and take $L = 9$. If the step size is Δ , then the decision levels are given as $d_i = -1 + i\Delta$, for $i = 0, 9$, so that $d_9 = -1 + 9\Delta$. Now this last value must equal $+1$, so we get $\Delta = 2/9$. (More generally we would have $\Delta = (x_{\max} - x_{\min})/L$.) Then from (9.3-1), we get

$$\begin{aligned}
 r_i &= \frac{1}{2}(d_i + d_{i-1}) \\
 &= -1 + \left(i - \frac{1}{2}\right)\Delta \\
 &= d_i - \frac{1}{2}\Delta.
 \end{aligned}$$

Note that there is an output at zero. It corresponds to $i = 5$, with $d_5 = +1/9$. This input bin, centered on zero, is commonly called the *deadzone* of the quantizer. ■

If a uniform quantizer is symmetrically located with respect to input value zero, then when the number of levels L is an odd number, we have a *midtread quantizer*, characterized by an input bin centered on 0, which is also an output value. Similarly, for such a uniform symmetric quantizer, when L is even, we have a *midrise quantizer*, characterized by a decision level at 0, and no zero output value.

Clearly, the MSE distortion of a quantizer will depend on the step sizes and on the total number of output levels L as well as upon the pdf f_x of the random variable x . For a uniform quantizer, this would translate into dependence on the step size Δ and number of levels L . A moment's consideration reveals that the quantizer MSE D should be proportional to the variance σ^2 of the pdf. In the case of *fine quantization* (i.e., many levels, closely spaced), we would expect that the pdf would be fairly constant across the decision intervals (bins), and so halving Δ , and a doubling of L to keep the range the same, should lead to one-quarter the MSE. We thus expect *quantizer models* of the form

$$\begin{aligned} D(\Delta) &= \kappa_1 \sigma^2 \Delta^2 \\ &= \kappa_1 \sigma^2 / L^2, \end{aligned}$$

where κ_1 and κ_2 are constants of proportionality that would depend on the pdf f_x . Further, if we were to encode each of the L output values as a fixed-length codeword, it would require on the order of $b = \log_2 L$ bits, so we can also expect that at least for fixed-length quantization, the MSE as a function number of bits b can be usefully modeled as

$$D(b) = \kappa \sigma^2 2^{-2b},$$

in the high rate or fine quantization case.

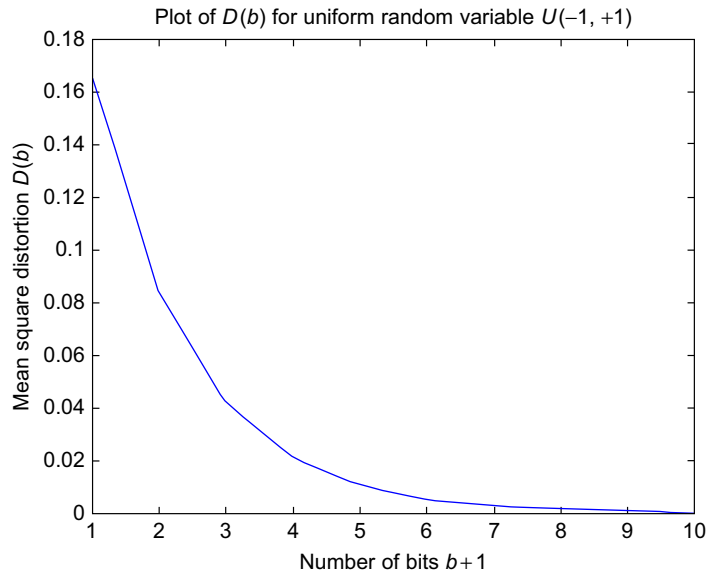
Example 9.3–2: Uniform Quantization of a Uniform Random Variable

In this example apply the uniform quantizer to a random variable that is uniformly distributed $U[-1, +1]$. The number of levels is chosen as $L = 2^b$ for the range $0 \leq b \leq 9$ or L range 1 to 512. The results are shown in Figure 9.3–3. The computation was done using the MATLAB program `UniUniformQuant.m` available at this book's Web site. Figure 9.3–4 is a plot of $\log_2 D(b)$ that displays a straight line of approximate slope -2 .

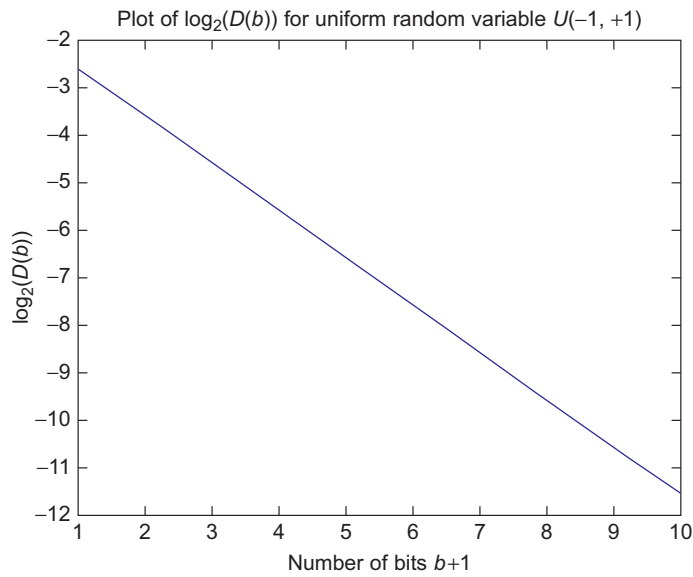
Optimal MSE Quantization

The average MSE of the quantizer output can be expressed as a function of the L output values and the $L - 1$ decision levels. We have, in the real-valued case,

$$\begin{aligned} D &= \int_{-\infty}^{+\infty} d^2(x, \hat{x}) f_x(x) dx \\ &= \sum_{i=1}^L \int_{d_{i-1}}^{d_i} (x - r_i)^2 f_x(x) dx \\ &\triangleq g(r_1, r_2, \dots, r_L; d_1, d_2, \dots, d_{L-1}). \end{aligned}$$

**FIGURE 9.3-3**

Plot of MSE of uniform quantization of uniform random variable $U[-1, +1]$ versus number of bits b .

**FIGURE 9.3-4**

Plot of $\log_2 D(b)$ versus b for uniform quantization of uniform random variable $U[-1, +1]$.

We can optimize this function by taking the partial derivatives with respect to r_i to obtain

$$\begin{aligned}\frac{\partial D}{\partial r_i} &= \int_{d_{i-1}}^{d_i} \frac{\partial}{\partial r_i} (x - r_i)^2 f_x(x) dx \\ &= \int_{d_{i-1}}^{d_i} 2(x - r_i) f_x(x) dx.\end{aligned}$$

Setting these equations to zero, we get a necessary condition for representation level r_i :

$$r_i = \frac{\int_{d_{i-1}}^{d_i} x f_x(x) dx}{\int_{d_{i-1}}^{d_i} f_x(x) dx} = E[x | d_{i-1} < x \leq d_i], \quad \text{for } 1 \leq i \leq L.$$

This condition is very reasonable; we simply set the output (representation) value for the bin $(d_{i-1}, d_i]$ equal to the conditional mean of the random variable x , given that it is in this bin. Taking the partial derivatives with respect to the d_i , we obtain

$$\begin{aligned}\frac{\partial D}{\partial d_i} &= \frac{\partial}{\partial d_i} \left\{ \int_{d_{i-1}}^{d_i} (x - r_i)^2 f_x(x) dx + \int_{d_i}^{d_{i+1}} (x - r_{i+1})^2 f_x(x) dx \right\} \\ &= (d_i - r_i)^2 p_x(d_i) - (d_i - r_{i+1})^2 p_x(d_i).\end{aligned}$$

Setting this equation to zero and assuming that $f_x(d_i) \neq 0$, we obtain the relation

$$d_i = \frac{1}{2}(r_i + r_{i+1}), \quad \text{for } 1 \leq i \leq L-1,$$

remembering that d_0 and d_L are fixed at the range of the input pdf support. This equation gives the necessary condition that the optimal SQ decision points must be at the arithmetic average of the two neighboring representation values. This is somewhat less obvious, but can be seen as picking the output value nearest to input x , certainly necessary for optimality.

An SQ Design Algorithm

The following algorithm makes use of these necessary equations to iteratively arrive at the optimal MSE quantizer. It is experimentally found to converge. We assume the pdf has infinite support here.

1. Given L , and the probability density function $f_x(x)$, we set $d_0 = -\infty$, $d_L = +\infty$, and set index $i = 1$. We make a guess for r_1 .
2. Use $r_i = \frac{\int_{d_{i-1}}^{d_i} x f_x(x) dx}{\int_{d_{i-1}}^{d_i} f_x(x) dx}$ to find d_i by integrating forward from d_{i-1} until a match is obtained.
3. Use $r_{i+1} = 2d_i - r_i$ to find r_{i+1} .
4. Set $i \leftarrow i + 1$ and go back to step 2, unless $i = L$.
5. At $i = L$, check

$$r_L \leq \frac{\int_{d_{L-1}}^{\infty} x f_x(x) dx}{\int_{d_{L-1}}^{\infty} f_x(x) dx}.$$

If “ $r_L >$ ”, then reduce initial value r_1 . Otherwise, increase r_1 . Then return to step 2. Continue this until convergence.

The amount of increase/decrease in r_1 has to be empirically determined, but this relatively straightforward algorithm works well to find approximate values for the decision and representation levels. Note that numerical integration may be needed in step 2, depending on the pdf f_x . An alternative algorithm is the scalar version of the Linde, Buzo, and Gray (LBG) algorithm for vector quantization, to be shown next.

Vector Quantization

Information theory says that quantizing signal samples one at a time (i.e., scalar quantization) can never be optimal [1, 5]. Certainly this is clear if the successive samples are correlated or otherwise dependent, and, in fact, that is the main reason for the transformation in our generic source compression system. But even in the case where the successive samples are independent, scalar quantization is still theoretically not the best way [1, 5], although the expected gain would certainly be much less in this case. *Vector quantization* (VQ) addresses the nonindependent data problem by quantizing a group of data samples simultaneously [2]. As such, it is much more complicated than scalar quantization, with the amount of computation increasing exponentially in the vector size, with practical VQ sizes limited to 4×4 and below, although some experimental work has been done for 8×8 blocksize [6]. Another difference is that VQ is typically designed from a training set, while SQ is typically designed from a probability model. This difference gives a significant advantage to VQ in that it can take use the “real” multidimensional pdf of the data, even in the absence of any viable theoretical multidimensional pdf model, other than joint Gaussian. As such, VQ coding results can be much better than those of SQ. In this regard, it should be noted that via its internal multidimensional pdf, the VQ effectively removes all dependencies in the data vector, and not just the so-called linear dependencies dealt with via a linear transformation. The downside, of course, is that 4×4 is a very small block. As a consequence, VQ is not often used alone without a transformation, but the two together can be quite powerful [7].

We later present a data-based design algorithm for VQ generally attributed to Linde, Buzo, and Gray (LBG), but we start out assuming that the theoretical multidimensional pdf is known. We consider an $N \times N$ random vector \mathbf{x} with joint pdf $f_{\mathbf{x}}(\mathbf{x})$

with support \mathcal{X} . Then we have a set of regions \mathcal{C}_l in \mathcal{X} that decompose this vector space in a mutually exclusive and collectively exhaustive manner—i.e.,

$$\mathcal{X} = \bigcup_{l=1}^L \mathcal{C}_l \text{ and } \mathcal{C}_l \cap \mathcal{C}_m = \phi, \text{ for all } l \neq m,$$

where ϕ is the null set.

On defining a set of *representation vectors* $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L\}$, the vector quantizer then works as follows

$$\hat{\mathbf{x}} = Q(\mathbf{x}) \triangleq \begin{cases} \mathbf{r}_1, & \mathbf{x} \in \mathcal{C}_1, \\ \vdots & \vdots \\ \mathbf{r}_L, & \mathbf{x} \in \mathcal{C}_L. \end{cases} \quad (9.3-2)$$

So in VQ the input-space *decision regions* \mathcal{C}_l play the role of the input decision intervals (bins) in SQ. Unfortunately, these input regions are much harder to define than simple bins, and this makes the actual quantizing operation $Q(\mathbf{x})$ computationally difficult. In order to perform the mapping (9.3-2), we have to find what region \mathcal{C}_l contains \mathbf{x} .

As was the case for scalar quantization, we can write the VQ error $\mathbf{e}_Q \triangleq \hat{\mathbf{x}} - \mathbf{x}$, and express the total MSE as

$$\begin{aligned} D &= E[(\hat{\mathbf{x}} - \mathbf{x})^T (\hat{\mathbf{x}} - \mathbf{x})] \\ &= \int_{-\infty}^{+\infty} d^2(\mathbf{x}, \hat{\mathbf{x}}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \\ &= \sum_{l=1}^L \int_{\mathcal{C}_l} d^2(\mathbf{x}, \mathbf{r}_l) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

One can show the following two necessary conditions for optimality:

- *Optimality condition 1* (useful for design of input regions \mathcal{C}_l and for actually quantizing the data \mathbf{x}):

$$Q(\mathbf{x}) = \mathbf{r}_l \text{ iff } d(\mathbf{x}, \mathbf{r}_l) \leq d(\mathbf{x}, \mathbf{r}_m), \text{ for all } l \neq m.$$

- *Optimality condition 2* (useful for determining the representation vectors \mathbf{r}_l):

$$\mathbf{r}_l = E[\mathbf{x} | \mathbf{x} \in \mathcal{C}_l]. \quad (9.3-3)$$

The first optimality condition states that, for a given set of representation vectors $\{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L\}$, the corresponding decision regions should decompose the input space in such a way as to cluster each input vector \mathbf{x} to its nearest representation vector in the sense of Euclidean distance $d(\mathbf{x}, \mathbf{r}_l)$. The second optimality condition says that if the input decision regions \mathcal{C}_l are given, then the best choice for their representation vector is their conditional mean.

From the first optimality condition, we can write

$$\mathcal{C}_l = \{\mathbf{x} | d(\mathbf{x}, \mathbf{r}_l) \leq d(\mathbf{x}, \mathbf{r}_m), \text{ for all } l \neq m\},$$

and such disjoint sets are called *Voronoi regions* [2]. So the large part of designing a VQ is to carve up the input space into these Voronoi regions, each determined by its property of being closer to its own representation vector \mathbf{r}_l than to any other.

Now this is all well and good, but the fact remains that we have no suitable theoretical joint pdf's $f_{\mathbf{x}}(\mathbf{x})$, other than multidimensional Gaussian, with which to calculate the conditional means in (9.3–3). However, if we were given access to a large set of input vectors $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$, all drawn from the same distribution, called the *training set*, the following data-based algorithm has been found to converge to a local minimum of the total square error over this training set [2], i.e., to provide a *least-squares solution* to VQ, with error metric

$$\sum_{i=1}^M d^2[\mathbf{x}_i, \mathbf{r}_l(\mathbf{x}_i)], \quad (9.3-4)$$

where $\mathbf{r}_l(\mathbf{x}_i)$ is just the VQ output for training vector \mathbf{x}_i .

LBG Algorithm [2]³

1. Initially guess the L output representation vectors $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L$.
2. Use optimality condition 1 to quantize each input vector in the training set; that is, find the vector \mathbf{r}_l that minimizes the distance $d(\mathbf{x}_i, \mathbf{r}_l)$. When this step finishes, we have partitioned the training set into the initial decision regions \mathcal{C}_l .
3. Use optimality condition 2 to update the representation vectors as

$$\mathbf{r}_l = \frac{1}{M_l} \sum_{\mathbf{x}_j \in \mathcal{C}_l} \mathbf{x}_j,$$

where $M_l \triangleq |\{\mathbf{x} | \mathbf{x}_j \in \mathcal{C}_l\}|$, denoting the number of training vectors in \mathcal{C}_l .

4. Go back to step 2 and iterate until convergence.

A bit of consideration of the matter reveals that steps 2 and 3 can only improve the optimality—i.e., reduce the error (9.3–4)—and never cause an increase in error. Hence the LBG algorithm converges to at least a local minimum of the total least-squares error. After design, the resulting VQ can be tested on a separate set of input data called the *test set*, that is assumed to have the same or similar joint distribution function.

Example 9.3–3: Vector Quantization

Consider a simple example with $M = 5$ training vectors,

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_5 = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

³For comparison, also see the K-means algorithm of Section 7.3.

to be vector quantized to $L = 3$ output vectors, $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$, using the LBG algorithm. We start with the initial guess

$$\mathbf{r}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{r}_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix},$$

and proceed into iteration one of the LBG algorithm. In step 2, we quantize (classify) the training data as follows:

$$\mathbf{x}_1 \rightarrow \mathbf{r}_2, \mathbf{x}_2 \rightarrow \mathbf{r}_1, \mathbf{x}_3 \rightarrow \mathbf{r}_2, \mathbf{x}_4 \rightarrow \mathbf{r}_3, \text{ and } \mathbf{x}_5 \rightarrow \mathbf{r}_1.$$

Then in step 3, we update the three representation vectors as,

$$\mathbf{r}_1 = \frac{1}{2}(\mathbf{x}_2 + \mathbf{x}_5) = \begin{bmatrix} 0 \\ -0.5 \end{bmatrix}, \mathbf{r}_2 = \frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_3) = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix},$$

$$\text{and } \mathbf{r}_3 = \mathbf{x}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

We then proceed into iteration two of the LBG algorithm. In step 2 we get the mapping

$$\mathbf{x}_1 \rightarrow \mathbf{r}_2, \mathbf{x}_2 \rightarrow \mathbf{r}_1, \mathbf{x}_3 \rightarrow \mathbf{r}_2, \mathbf{x}_4 \rightarrow \mathbf{r}_3, \text{ and } \mathbf{x}_5 \rightarrow \mathbf{r}_1,$$

which is unchanged from step 2 of iteration one. Thus the second, or update, step will leave the representation vectors \mathbf{r}_i unchanged from those of the first iteration. So we are converged at iteration two, with representation vectors

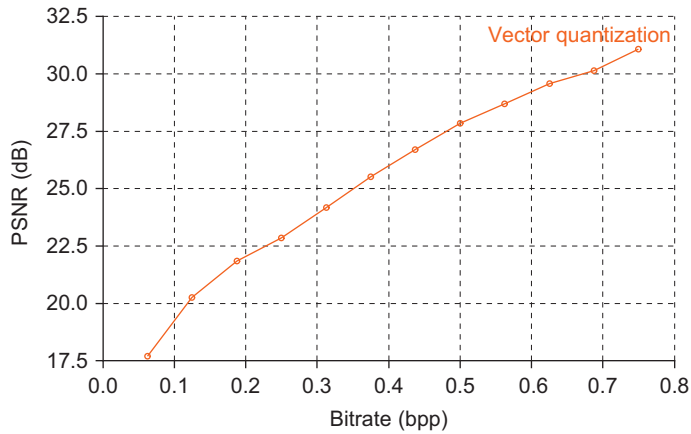
$$\mathbf{r}_1 = \begin{bmatrix} 0 \\ -0.5 \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}, \mathbf{r}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

It is interesting to note that the optimality is only local for the LBG algorithm, meaning the convergence point depends on the initial guess. For example, if in this example we had started with the initial guess $\mathbf{r}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{r}_3 = \begin{bmatrix} 1/2 \\ -1 \end{bmatrix}$, then LBG would converge to a different VQ—i.e., different set of representation vectors (see problem 9.7). For this reason, users often try several starting points to ensure a good minimum.

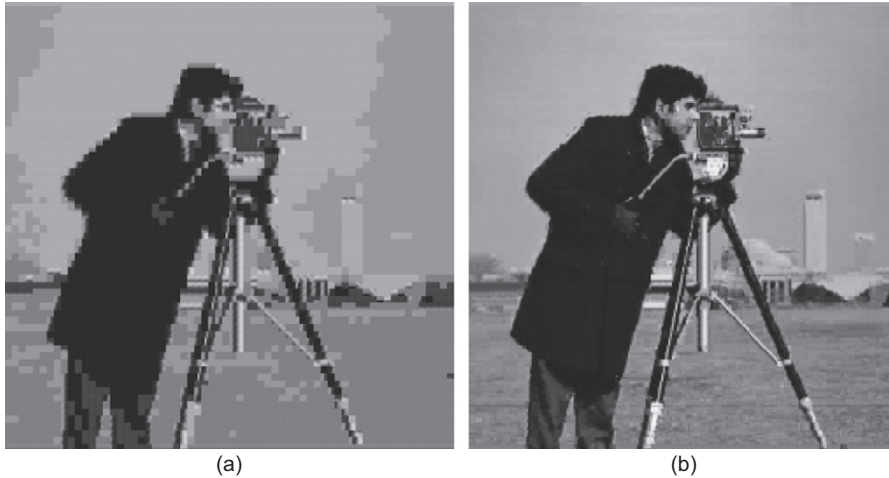
VQ is not used in today's image coding standards, but the related technique trellis-coded quantization (TCQ) appears as an optional part in the JPEG 2000 standard [8]. We will discuss the basic part of this image compression standard in [Section 9.7](#).

Example 9.3–4: VQ on *Cameraman*

Here, we perform LBG VQ on the 256×256 gray-level cameraman image. We used the freely available software *VcDemo* [9] to form 4×4 codebooks with random initialization, for 1–12 bits per vector, fixed-length codewords. The resulting PSNR versus bits/pixel is

**FIGURE 9.3-5**

Plot of PSNR versus bits for 4×4 VQ on the cameraman image.

**FIGURE 9.3-6**

LBG VQ 4×4 result on the cameraman image: (a) 4 bits/vector, (b) 8 bits/vector.

plotted in Figure 9.3-5 as produced by *VcDemo*. We also show 2 of these 12 results in Figure 9.3-6. The image on the left is made from 16 4×4 vectors (i.e., 4 bits/vector), and the one on the right is made from 256 4×4 vectors (i.e., 8 bits per vector). They are, respectively, 1/4 bits per pixel (bpp) and 1/2 bpp. The quantized image with $16 = 2^4$ vectors shows poor quality, while the one made from $256 = 2^8$ vectors begins to look acceptable, at least for some uses. ■

9.4 ENTROPY CODING⁴

After the quantization, we must send the message set of quantizer outputs to the channel or channel coder. Without loss of generality, we can think in terms of binary digits or bits for this information.⁵ In general we can use a fixed-length or variable-length code for this purpose. Now, the mathematical theory of communication defines *information* [1, 5] quantitatively for an independent, discrete-valued, stationary random source as

$$I(x_i) \triangleq \log_2 \frac{1}{p(x_i)},$$

and its average value *entropy* as

$$H(X) = \sum_i p(x_i) \log_2 \frac{1}{p(x_i)}. \quad (9.4-1)$$

It is proved in [1, 5] that any coding of such a source into binary digits must have an average codeword length greater than or equal to this entropy. The so-called *source coding theorem* states that, for the encoding of one message at a time,

$$H(X) \leq \bar{l} < H(X) + 1, \quad (9.4-2)$$

where \bar{l} denotes the average codeword length $\bar{l} \triangleq \sum_i p(x_i)l(x_i)$. If we were to go to the expense of jointly coding M messages simultaneously, theory says that this bound becomes

$$H(X) \leq \bar{l} < H(X) + \frac{1}{M},$$

due to the assumed independence of the random source. Unfortunately, such coding of the so-called *extension source* is not often practical. In both these equations, H is the entropy per single message.

Huffman Coding

In the 1950s, David Huffman came up with an optimal variable-length encoding procedure, which came to be known as *Huffman coding* [1, 5]. It is a method for coding a finite number of messages into variable-length binary codewords in such a way that the average codeword length is minimized. Huffman coding creates a binary tree by first joining together the two messages with the lowest probabilities, thereby creating a reduced source with one fewer messages. It then proceeds recursively to join together the two messages with lowest probability for this reduced source, proceeding

⁴This section requires a background in information theory. An introduction to information is provided in the appendix to this chapter.

⁵This is the so-called quantizer *index set*. We assume that the receiver knows the quantizer design and can map the set of quantizer indices back into the actual representation values. Such a mapping is often given the misnomer *inverse quantization*, and in practice would be communicated to the receiver as the default values in an international standard or perhaps as header information in a packet or file.

down to a root node whose probability is one. When the branches of this tree are populated with ones and zeros, a variable length code can be read off in the reverse order. This is perhaps best illustrated by example.

Example 9.4–1: Huffman Coding

Let the source have $M = 5$ messages, with probabilities $p_1 = 0.5$, $p_2 = 0.3$, $p_3 = 0.1$, $p_4 = 0.05$, $p_5 = 0.05$. Then we can construct the Huffman code as shown in Figure 9.4–1. Here, starting from the left, messages 4 and 5 have the lowest probabilities and are therefore combined first. Their total probability is then 0.1. It happens that this node is also a smallest probability in the reduced source, and it is joined with message 3 with $p(3) = 0.1$ at the second level. We continue on to complete the tree. The variable-length codewords are now read from the right and become, reading from message 1 to message 5: 0, 10, 110, 1110, and 1111, with lengths $l_i = 1, 2, 3, 4$, and 4, respectively, in this example. We note that this variable-length code is uniquely decodable, because no codeword is the prefix of another codeword. It can be seen that the Huffman code tree guarantees that this will always be true. Computing the average codeword length, we get

$$\begin{aligned}\bar{l} &\triangleq \sum_i p(x_i) l(x_i) \\ &= 0.5 \times 1 + 0.3 \times 2 + 0.1 \times 3 + 0.05 \times 4 + 0.05 \times 4 \\ &= 1.8 \text{ bits/message.}\end{aligned}$$

In this case the entropy from (9.4–1) is $H = 1.786$. We note that the average codeword length for this example is quite close to the entropy of the source. This is not always the case, especially in the case when the entropy per message is much less than one, as forewarned in (9.4–2).

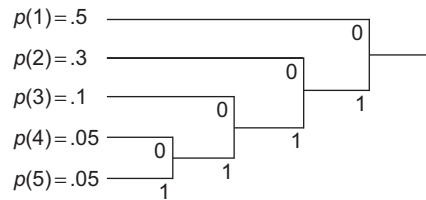


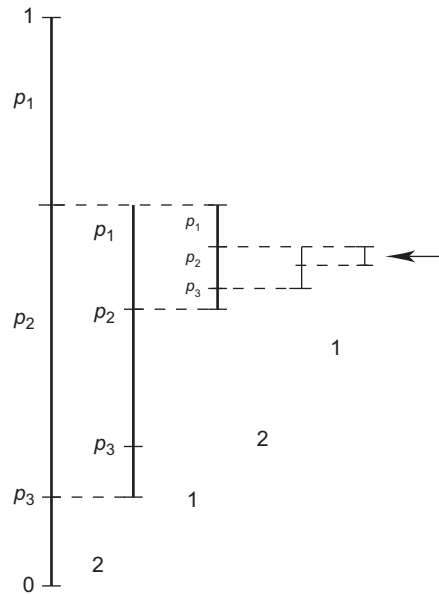
FIGURE 9.4–1

Example of Huffman coding tree.

In passing, we note the the Huffman variable-length tree is complete, i.e., all branches terminate in valid messages. Hence, the resulting code stream will be sensitive to channel errors. More on this later.

Arithmetic Coding

The Huffman coding procedure codes a fixed number of messages, usually one, into variable-length binary strings. Another popular coding method, in contrast, codes a

**FIGURE 9.4-2**

An essential aspect of arithmetic coding.

variable number of messages into fixed-length binary strings. This method, called *arithmetic coding* (AC), can usually achieve a very close match to the entropy of the source. AC works with subintervals of $[0, 1]$, with lengths equal to the probabilities of the messages to be coded.

As illustrated in Figure 9.4-2, as messages are processed, the subinterval is successively split according to the message probabilities. First, in this example, we “send” message 2 by centering attention on the middle interval of length p_2 . Then we send message 1 by centering attention on the subinterval p_1 . We then “send” the following messages 2 and 1 by the same approach. This continues until a fixed, very small subinterval size is reached, or is about to be exceeded. If we “convey” the final subinterval to the receiver, then the decoder can decode the string of messages that lead up to this final subinterval of $[0, 1]$. We note in passing that there is nothing to prevent the message probabilities from changing in some prescribed manner as the successive messages are “encoded” into subintervals. The final subinterval is actually conveyed to the receiver by sending a fixed-length binary string of sufficient number of digits to point uniquely to the pre-chosen fixed but small subinterval. This is indicated by the pointer in Figure 9.4-2. A typical pointer value is 14 bits or so. Note that decoding can proceed while the data are transmitted in a first-in, first-out fashion, because as the subinterval shrinks in size, the leading digits of the fixed-length binary string are successively determined. A lot of practical details are left out of the simple argument here, including the important question of numerical significance for a fixed-length computer word implementation; however, the basic idea is as shown.

The question remains, how efficient is this method? To provide a crude answer to this question, we note that the final subinterval size will be the product of the message probabilities that are encoded $\prod_i p(x_i)$, and that the approximate number of binary digits to uniquely point to such an interval is

$$L = -\log_2(2^{-L}) \simeq -\log_2[\prod_i p(x_i)] = \sum_i -\log_2[p(x_i)],$$

where the term on the right is the sample total, the average value of which is the entropy of this independent string of messages from the source. By the law of large numbers of probability theory, for an ergodic source, we can expect convergence to the entropy (9.4–1). So we can expect to get very close to the entropy of a stationary ergodic source in this way. A more thorough derivation of AC, including algorithms, can be found in Sayood [10] and Gersho and Gray [2].

ECSQ and ECVQ

It is more efficient to combine quantization and entropy coding into one operation. *Entropy-coded scalar quantization* (ECSQ) and *entropy-coded vector quantization* (ECVQ) accomplish just that. For a nonstructured VQ(SQ) we can start with the VQ design problem of Section 9.3 (Vector Quantization) and augment the error criteria with the Lagrange cost as

$$\begin{aligned} E &= D + \lambda H(R), \\ &= \sum_{l=1}^L \int_{C_l} d^2(\mathbf{x}, \mathbf{r}_l) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} + \lambda H(R), \end{aligned}$$

where $H(R)$ is the resulting entropy of the quantized source X with representation vectors \mathbf{r}_l , and λ is a Lagrange multiplier. The parameter λ then becomes a variable controlling a tradeoff of rate $H(R)$ versus distortion D . Here, the entropy is computed as

$$H(R) = \sum_{l=1}^L p_l \log_2 \frac{1}{p_l},$$

with p_l being the probability mass function (pmf) of the quantizer outputs \mathbf{r}_l .

Experimentally, we can use the LBG algorithm on a training set as in Section 9.3 (LBG Algorithm), just augmenting the error (9.3–4) with lambda times the experimental entropy computed from a histogram over the training vectors. To get the complete distortion-rate function of the resulting ECVQ(ECSQ) system, we merely run this modified LBG algorithm for a range of λ values and plot the resulting points. These points can then be connected with straight lines to get the final estimate of the joint quantizer-entropy model. Since conditional AC has been found to be extremely efficient for image-coding applications, one can effectively take the resulting rate-distortion characteristic and regard it as a virtually joint quantizer-AC model also.

Taking a structured SQ, such as a UTQ with stepsize Δ and deadzone 2Δ , and number of levels L , we can plot the rate-distortion characteristics of an ECSQ directly

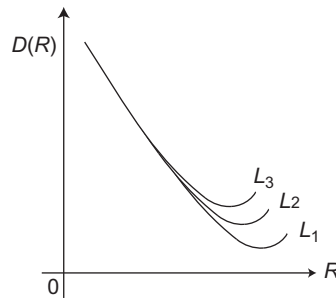
**FIGURE 9.4-3**

Illustration of ECSQ joint quantizer and entropy encoder.

as a function of parameter Δ , typically getting plots such as shown in Figure 9.4-3. We see that large values of L do not penalize results below a certain maximum rate, at which point saturation begins to occur. This is mainly a consequence of the fact that messages with very small probability do not add much to the entropy of a source, via $\epsilon \log_2 \epsilon \searrow 0$ as $\epsilon \searrow 0$. By making sure to operate at lower rates, quantizer saturation can be pushed out of the range by using a large number of levels, without hurting the lower rate performance. More on ECVQ and ECSQ is contained in the paper by Kim and Modestino [11].

Error Sensitivity

Both Huffman and arithmetic coding are examples of *variable-length coding* (VLC). In any coding system that uses VLC, there is an increased error sensitivity because a bit error will probably cause loss of synchronization, with resultant incorrect decoding until a fortuitous chance event occurs later in the bitstream, i.e., until by chance, a correct decoding occurs. This is not the case in fixed-length coding. As a result, the extra compression efficiency that can be obtained using VLC leads to increased sensitivity to errors. In practice, sync words and limitation on the length of VLC bitstreams can effectively deal with this increased sensitivity problem, but only with added complexity. A VLC coder that does not have such error-resilience features is not really practical, as a single bit error anywhere in the image would most likely terminate correct decoding for the rest of the image. So, all but the most pristine channels or storage media will need some kind of error-resilience features. We talk more about this in Chapter 13 for the image sequence or video case.

9.5 DCT CODER

DCT coders, the most representative of these being the international standard JPEG, use the tools presented here. The input image frame is first split into its Y , U , and V components. The Y , or luma, component is then subjected to block-DCT with nonoverlapping 8×8 blocks. After this the DCT coefficients are split into one DC and 63 AC coefficients. They are then quantized by scalar uniform quantizers, with

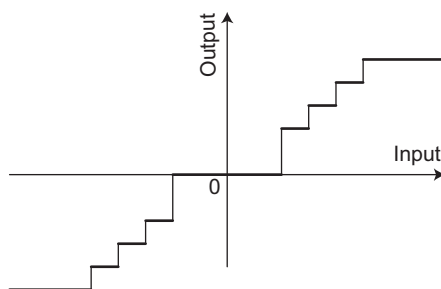
**FIGURE 9.5-1**

Illustration of UTQ. Note its central deadzone.

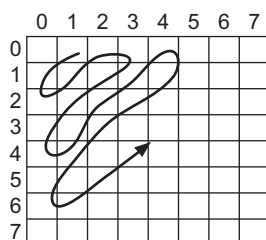
**FIGURE 9.5-2**

Illustration of zigzag or serpentine scan of AC coefficients of 8×8 DCT.

possibly a central deadzone, with step size determined by a *quantization matrix* that gives a preassigned step size to each of the 64 DCT coefficients. A quantizer with a central dead zone is called a uniform threshold quantizer (UTQ) as seen in Figure 9.5-1. It can be specified by its number of levels L , step size Δ , and the width of its deadzone.⁶ After quantization, the DC coefficient may undergo lossless DPCM coding across the blocks, while the AC coefficients are scanned in a certain zigzag or serpentine scan (seen in Figure 9.5-2) and input to a 2-D *run-value* or 2-D *Huffman code*. This code assigns codewords to pairs of messages indicating a run of xx zeros terminated by a nonzero DCT value yy . This 2-D VLC is stored in a table in the coder and the decoder.

There remains the problem of bit assignment to the DCT coefficients. This can be done as follows. We first assume a quantizer model for the DCT coefficient with index k_1, k_2 , say

$$D_{k_1, k_2} = g \sigma_{k_1, k_2}^2 2^{-2R_{k_1, k_2}}, \quad (9.5-1)$$

⁶The central deadzone is often useful for discriminating against noise in the coefficients to be quantized. Also, a central deadzone of 2Δ arises naturally in so-called bit-plane coding in embedded scalable coding (see Section 9.6).

where σ_{k_1,k_2}^2 is the coefficient variance (we assume zero mean) and R_{k_1,k_2} is the number of bits assigned to this coefficient. Due to orthogonality of the unitary DCT, the total MSE in the reconstruction will be

$$D = \sum_{k_1,k_2} D_{k_1,k_2},$$

and the average bitrate will be

$$R = \frac{1}{N^2} \sum_{k_1,k_2} R_{k_1,k_2},$$

for an $N \times N$ DCT. To minimize the MSE, we introduce the Lagrangian parameter λ and seek to minimize the function

$$D + \lambda R = \sum_{k_1,k_2} D_{k_1,k_2} + \lambda \left(R - \sum_{k_1,k_2} R_{k_1,k_2} \right).$$

Taking the partial derivatives with respect to each of R_{k_1,k_2} and setting them to zero, we can solve for the parameter λ and eventually obtain the following solution:

$$R_{k_1,k_2} = R + \frac{1}{2} \log_2(\sigma_{k_1,k_2}^2 / \sigma_{\text{wgm}}^2), \quad (9.5-2)$$

with

$$\sigma_{\text{wgm}}^2 \triangleq \left(\prod \sigma_{k_1,k_2}^2 \right)^{1/N^2}.$$

The resulting total MSE for the N^2 -pixel block then becomes simply

$$D = N^2 g \sigma_{\text{wgm}}^2 2^{-2R}. \quad (9.5-3)$$

It is surprising that the result only depends on the average bits/pixel R and the geometric mean of the coefficient variances σ_{wgm}^2 . There is some variation in MSE across the $N \times N$ block due to the fact that pixels near the block boundaries do not have as many neighbors to help reduce the coding error as do those in the middle of the block. This effect is not large, though.

Example 9.5–1: Block-DCT Coding

The 252×256 monochrome cameraman image was coded by the DCT coder in *VcDemo* [9] at approximately 1 bpp. An 8×8 block-DCT was used, with DPCM coding of the DC coefficient from block-to-block, and PCM coding of the AC coefficients. Then VLC was applied to the quantized coefficient indices. The original image is shown in Figure 9.5–3 and the coded image at actual bitrate 0.90 bpp is shown in Figure 9.5–4.

The original block-DCT coefficients are shown in Figure 9.5–5, and the quantized coefficients are shown in Figure 9.5–6. We can see that about half of the DCT coefficients have

**FIGURE 9.5-3**

Original 252×256 cameraman image.

**FIGURE 9.5-4**

DCT-coded image at 0.90 bpp.

been set to zero, yet the coded image still looks quite good. A close-up of [Figure 9.5-4](#), though, reveals coding distortion, especially around edges in the image. The PSNR is 29.4 dB.

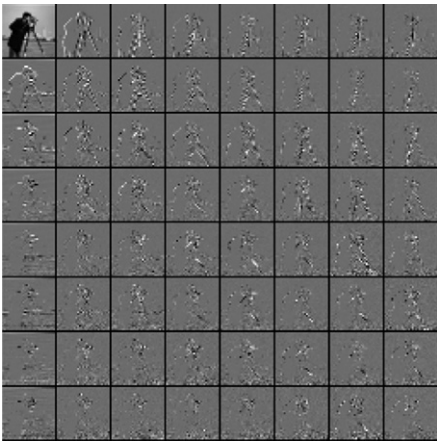


FIGURE 9.5-5
DCT coefficients of the cameraman image.

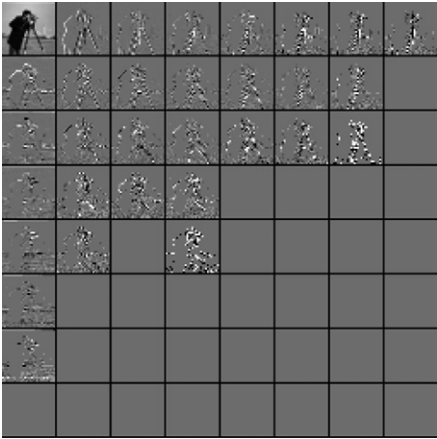


FIGURE 9.5-6
Quantized DCT coefficients of the cameraman image.

9.6 SWT CODER

Due to their advantages of no blocking artifacts and somewhat better compression efficiency, subband/wavelet coders have been extensively studied in recent years. Here, we consider the popular dyadic or recursive subband decomposition, also called the wavelet decomposition, which was mentioned previously. This transformation

is followed by scalar quantization of each coefficient. Rate assignment then splits up the total bits for the image into bit assignments for each subband. Assuming an orthogonal analysis/synthesis system is used, the *average* MSE distortion D due to the individual quantizations with distortion $D_m(R_m)$, with R_m bits assigned to the m th subband quantizer, can be written as the weighted sum of the mean-square distortions over a total of M subbands as

$$D = \sum_{m=1}^M \frac{N_m}{N} D_m(R_m), \quad (9.6-1)$$

where N_m is the number of samples in the m th subband, and the total number of pixels (samples) is N . Similarly, the average rate is given as

$$R = \sum_{m=1}^M \frac{N_m}{N} R_m.$$

Note that these weighted averages over the M subbands are the same as unweighted averages over the N subband samples.

If we model the individual quantizers as $D_m = g\sigma_m^2 2^{-2R_m}$, all with the same quantizer efficiency factor g , then the optimal bit assignment for an average bitrate of R bits/pixel, can be found with some assumptions, via the Lagrange multiplier method [8], to yield

$$R_m = R + \frac{1}{2} \log_2 \left(\frac{\sigma_m^2}{\sigma_{\text{wgm}}^2} \right), \quad m = 1, \dots, M, \quad (9.6-2)$$

where the weighted geometric mean σ_{wgm}^2 is defined as

$$\sigma_{\text{wgm}}^2 \triangleq \prod_{m=1}^M (\sigma_m^2)^{N_m/N},$$

where N is the total number of samples, i.e., $N = \sum_{m=1}^M N_m$. The resulting MSE in the reconstruction then becomes

$$D = g \sigma_{\text{wgm}}^2 2^{-2R},$$

which is analogous to (9.5-3) of DCT coding.

If we were to compare a DCT coder to a SWT coder, using this bit assignment method, all that would be necessary is to compare the corresponding weighted geometric means of the variances. Note that there are a number of assumptions here. First, we assume the quantizer model of (9.5-1), which is strictly only valid at high bitrates. Second, rate-distortion theory says this type of coding is only optimum for jointly Gaussian distributions. In practice, the quantizer model of (9.5-1) is only useful as a first approximation. More exact and complicated quantizer models have been developed for specific applications.

Example 9.6–1: Subband Coding

The original 512×512 *Lena* image of Figure 9.6–1 was coded by the subband coding module of *VcDemo* [9] at 1 bpp. Two levels of subband decomposition were obtained using 16-tap separable filters, yielding 16 subbands. The DC subband was quantized by DPCM using a 1×1 -order NSHP linear predictor, while the remaining AC subbands were coded by uniform quantization. VLC was then applied to the quantizer outputs. Bit assignment was done according to the preceding procedure and resulted in the bit assignments shown in Table 9.6–1, resulting in an overall average bitrate of 0.99 bpp. The result is shown in Figure 9.6–2 where the actual bits used are 0.95 bpp and the PSNR is reported to be 37.4 dB for a standard deviation error of 3.4 on this 8-bit image with range [0,255].



FIGURE 9.6–1

Original 512×125 Lena—8 bits.

Table 9.6–1 Bit assignments for 4×4 DCT with (0,0) in the upper left-hand corner			
4.64	3.41	2.74	1.81
1.81	0.00	1.50	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00

**FIGURE 9.6–2**

512 × 512 Lena coded by subband coding at 1 bpp.

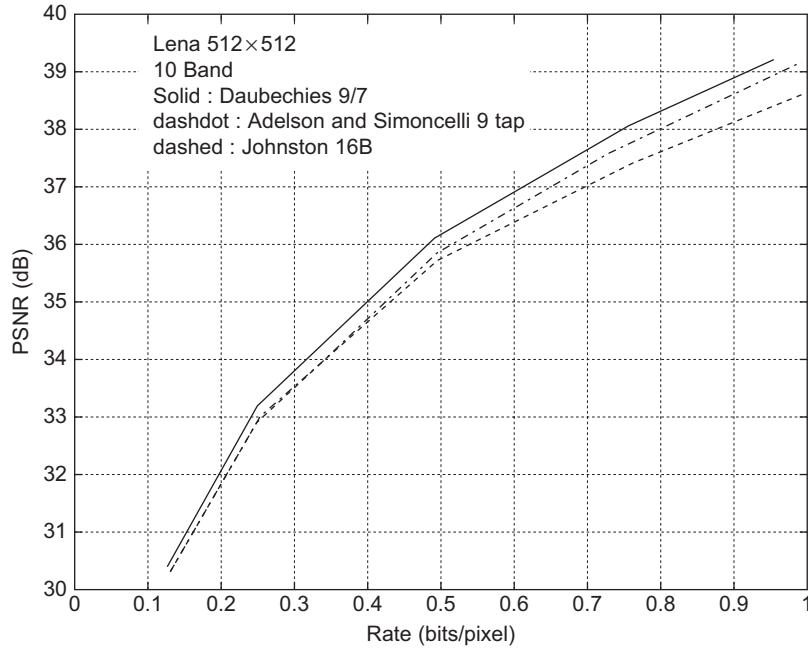
Example 9.6–2: SWT Coder Performance

An example SWT image coder is reported in Choi [12] where a filter study is undertaken to evaluate the performance of various subband/wavelet filters for compression of the 512 × 512 Lena image. We used the SB-FSSQ embedded coder [13] and compared three filters using a 10-band dyadic subband decomposition. The filters considered were Daubechies 9/7, Johnston's 16B, and Adelson and Simoncelli's 9 tap. The PSNR results in Figure 9.6–3 show quite close performance with the top-performing PSNR curve corresponding to the Daubechies 9/7 filter. Interestingly enough, the 16B filter came out on top when we coded all 16 subbands rather than taking a dyadic or wavelet decomposition. One additional comment is that the Daubechies 9/7 is said to have only a small amount of ringing distortion when the LL subband is displayed alone, as happens in scalable coding. In fact, a key property of subband/wavelet coders is their inherent scalability in resolution to match display and communications link bandwidth requirements.

A more precise approach to bit assignment may be done as follows. First we write the total rate as

$$R = \sum_{i=1}^N R_i,$$

where we assume N channels (or coefficients). We now wish to find the channel rate assignment $\mathbf{R} = (R_1, \dots, R_N)^T$, such that the total distortion in (9.6–1) is minimized.

**FIGURE 9.6-3**

Comparison of three subband/wavelet filters on the Lena image.

Effectively we seek the

$$\mathbf{R} \triangleq \arg \min_{\mathbf{R}} \sum_{i=1}^N D_i(R_i).$$

This problem of minimization with constraint can be formulated with a Lagrange multiplier as follows: $D(\mathbf{R}) \triangleq \sum_{i=1}^N D_i(R_i)$, and then

$$f(\mathbf{R}) = D(\mathbf{R}) + \lambda \left(R - \sum_{i=1}^N R_i \right).$$

Taking partial derivatives with respect to each R_i , we obtain the so-called constant slope conditions

$$\frac{\partial f}{\partial R_k} = \frac{\partial D_i}{\partial R_i} - \lambda = 0,$$

meaning that at optimality we should operate at points on the channel distortion-rate curves that all have the same slope,

$$\frac{\partial D_i}{\partial R_i} = \lambda \quad (9.6-3)$$

known as the *equal slope condition*. As we vary the value λ , we then can sweep out the entire distortion-rate curve $D(R)$.

Example 9.6–3: Discrete Quantizer Set

Often in practice we only have a discrete number of bit assignment points because a limited number of fixed quantizers can be used. Then there are no continuous distortion-rate curves $D_i(R_i)$ for the individual channels. In that case, if we try all the combinations of the various quantizers, we get a large but finite set of data, as sketched in Figure 9.6–4. We notice that the best points are to the left and downward, because for a given rate, these values have the lowest total distortion. This is called the *convex hull* of this set of data points. At the highest total rate, we just use the smallest step size available among all the quantizers. Likewise, the optimal lowest rate would be obtained by using the largest step size for each quantizer, sometimes even leaving that channel(s) out altogether (i.e., assigning it zero bits). A method to solve for the intervening points based on tree optimization is the BFOS algorithm [14], which is discussed in a problem at the end of this chapter.

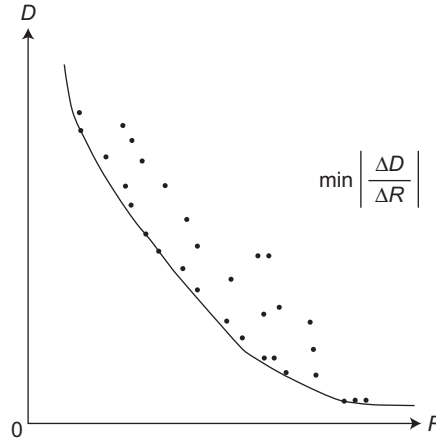


FIGURE 9.6–4

Illustration of $D(R)$ in the case of a discrete number of quantizer step sizes.

Multiresolution SWT Coding

Figure 9.2–2 shows a dyadic SWT with three recursive applications of the basic separable four-band SWT. We thus generate three lower resolution images. We call the first generated LL subband LL_1 and the second generated LL subband LL_2 , and so on. For completeness we call the original image LL_0 . Here, we consider only two levels of decomposition giving a lowest resolution of LL_2 , medium resolution of LL_1 , and full resolution LL_0 . A scalable coder, embedded in resolution, can then be constructed by first coding the lowest resolution data LL_2 , and then an enhancement band for the subbands included in LL_1 , and finally a further enhancement band consisting of the remaining subbands LH, HL, and HH, making up LL_0 . If we simply transmitted these three scales, LL_i , $i = 0, 3$, we would have what is called *simulcast*, an

inherently inefficient method of transmission. To make this scheme more efficient, a residual, or error, from the lower scale can be passed on to the next higher scale and coded. A diagram illustrating this residual coding approach is shown in Figure 9.6–5.

Following partition based on scale, the low resolution coder C0 is a conventional SWT image coder. The medium resolution coder C1 gets all the data up to the LL_2 or midspatial resolution. However, the coded error from C0 is also fed to the intermediate coder C1, which internally codes this error in place of the low resolution subbands in LL_2 , thereby providing a refinement of these data as well as coding with the new midfrequency data. This process is repeated again for coder C2. Decoding proceeds as follows. To decode low-resolution data, there is no change from conventional coding. To decode medium-resolution data, the delta1 enhancement layer is decoded and used to supplement the low-resolution layer. A similar method then uses bitstream delta2 to achieve the full spatial resolution.

One might ask why the coder error has to be propagated across the scales in a scalable coder; in other words, “Why not just code the high-frequency subbands ($-LH$, $-HL$, and $-HH$) at the new scale?” The reason is that the rate assignment (9.6–2) will give different numbers of bits to the LL_2 subband depending on whether it is part of the low-, the medium-, or the high-resolution image. This assigned number of bits will generally increase with resolution, if the desired PSNR is around the same number or is increasing with resolution, which is often the case. So recoding the coding error of the bands already transmitted effectively increases the bit assignment for that band. Focusing on the LL_2 subband, for instance, it is coded once, then recoded for the medium-resolution image, and then recoded a second time for the high- or full-resolution image. For a given subband component of the LL_2 image, we can see a concatenation of quantizers, as shown in Figure 9.6–6. The input signal is quantized at rate $R1$ to produce bitstream $b1$. The rate $\Delta 2$ is then used to refine the representation, with corresponding enhancement bitstream $b2$. The process is repeated again with enhancement rate $\Delta 3$ and corresponding bitstream $b3$. A decoder receiving only $b1$ can decode a coarse version of the input. With the addition of $b2$, a decoder can refine this coarse representation, with information about the initial quantization error. Reception of the second enhancement layer can then refine the representation

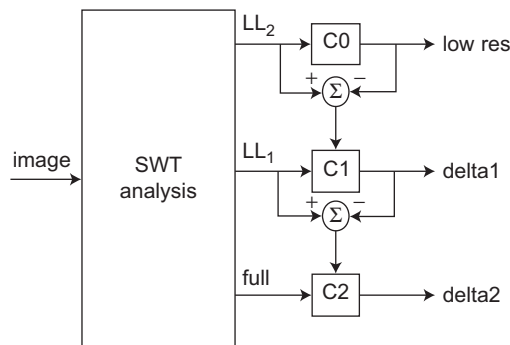
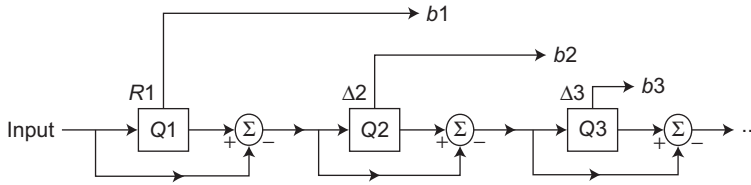


FIGURE 9.6–5

Multiresolution image coder with three decoded image resolutions (full, 1/2, and 1/4).

**FIGURE 9.6–6**

A cascade of three quantizers provides a quality scalable coding of input signal.

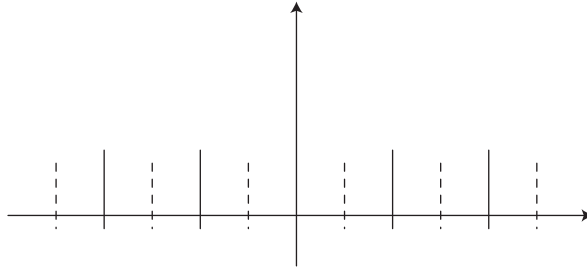
**FIGURE 9.6–7**

Illustration of scalar quantizer embedding. Solid lines are coarse quantizer decision levels. Dashed and solid lines together constitute decision levels of the combined embedded quantizer.

even more. This is then a rate-scalable way to transmit a pixel source, giving three possible bitrates.

Such quantizers are said to be embedded, because the successively higher accuracy quantizers defined in this way have the property that each stage refines the decision levels of the previous stage. This is illustrated in Figure 9.6–7, which shows the overall quantizer bins for two stages of embedding. Here, the solid vertical lines are decision levels for the coarse quantizer Q1 and the dashed lines are decision levels provided by Q2, thus yielding an equivalent quantizer consisting of all the decision levels.

In an embedded quantizer, the first stage can be a MSE optimal quantizer, but the second and succeeding stages are restricted to using these same decision levels plus some more, which would not normally be optimal. A key paper by Equitz and Cover [15] considered the optimality of such cascaded or refined quantizers and concluded that the source must have a certain Markov property holding across the refinements for optimality. In particular, successive refinement bits from the source must be conditionally dependent only on the next higher significance bit. In general, a refinement bit would be dependent on all the prior more significant bits. Still, the greatest dependency would be expected to occur on the next higher significance bit. If all of the quantizers are uniform and the step sizes are formed as decreasing powers of two (i.e., 2^{-k}), then the outputs b_k become just the successively smaller bits from a binary representation of the input value. In this case all of the quantizers are uniform threshold and are said to be *fully embedded*.

Nondyadic SWT Decompositions

As mentioned earlier, the dyadic or wavelet decomposition does not always give the best compression performance; so-called *wavelet packets* optimize over the subband tree, splitting or not, to get the best coding quality for that frame [4]. For the case when the splitting is allowed to change within an image (frame), they formulated a so-called *double tree algorithm* to find a spatially adaptive decomposition. Note, however, that this is not an easy problem, since the best subband/wavelet decomposition for a given image class will depend on the quantizing method used as well on the entropy coder.

Fully Embedded SWT Coders

This class of coders follows the SWT by a variable-length coder that first codes the most significant subband samples/coefficients at their most significant bit plane and then proceeds down in significance, bit plane by bit plane, both noting newly *significant* coefficients and also refining those coefficients already coded at higher bit planes. The generally acknowledged first of these was *embedded zero-tree wavelet* (EZW) by Shapiro [16]. The EZW coding algorithm was followed by the SPIHT coder of Said and Pearlman [17]. The coding standard JPEG 2000, which can be considered a kind of fruition of this type of coder, is covered thoroughly in the text by Taubman and Marcellin [8]. We also present the basic concept of the embedded zero block coder (EZBC) [18] that has gained interest in scalable video coding research.

A common diagram of the subband structure of the SWT coder is shown in Figure 9.6–8, which represents three stages of subband/wavelet decomposition. The first stage results in subbands LL_1 , LH_1 , HL_1 , and HH_1 . The second stage then decomposes LL_1 to yield subbands LL_2 , LH_2 , HL_2 , and HH_2 . Finally, there is a third stage of decomposition that results in subbands LL_3 , LH_3 , HL_3 , and HH_3 . Subband LL_3 is not further decomposed here and plays the role of *baseband*. We can then look at the sets of three high-frequency subbands $\{LH_k, HL_k, \text{ and } HH_k\}$ for $k = 3, 2, 1$, as enhancement data that permit an increasing scale (resolution) over the

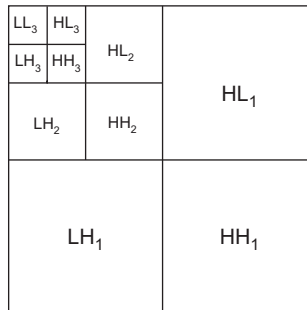


FIGURE 9.6–8

Illustration of subband/wavelet structure for the dyadic (also called octave band or wavelet) decomposition, where the subbands XY_k are at the k th stage.

coarse scale representation in the baseband. It is important to realize that this diagram is in the spatial domain, in that each subband is displayed spatially within each subwindow XY_k . Equivalently, we are looking at wavelet coefficients.

As already mentioned, embedded coders proceed to code the data of Figure 9.6–8 by proceeding from most significant bit (MSB) to least significant bit (LSB), bit plane by bit plane. Of course, this implies a quantization; in fact, it's a UTQ, specified by its step size Δ and deadzone centered on the origin and of width 2Δ . Assuming for the subband to be coded, that the LSB bit plane is 2^b , then $\Delta = 2^b$. This LSB bit plane is usually conservatively chosen at such a level that its quality will never need to be exceeded.

Embedded Zerotree Wavelet (EZW) Coder

The EZW bit-plane coder proceeds from one quite significant observation: at low and even into medium bitrates, there are a lot of zeros in all but the MSB plane, and once a zero is encountered in a bit plane at a given position, then for natural images it is very likely accompanied by zeros at higher frequency subbands corresponding to the same location. Shapiro defined a special symbol, called *zero-tree root* [16], to code all these zeros together. He regards the quantized data in Figure 9.6–8 as composing separate data trees, each with roots in the base subband LL_N (here $N = 3$), where N is the number of levels of decomposition. Figure 9.6–9 shows the *parent-child relationships* for the three trees of coefficients (subband samples) growing out of one spatial location in the LL_3 subband.

In EZW coding, the first step is to find the MSB plane by computing the maximum over all the subband magnitude values,

$$2^T \leq \max |c_{i,j}| < 2^{T+1},$$

for subband data $c(i,j)$.

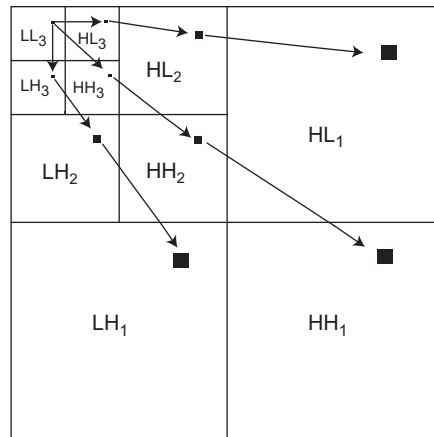


FIGURE 9.6–9

Illustration of parent-child relationship for trees of coefficients in EZW. Tree depth $N = 3$ levels.

Then we can normalize the data and introduce the binary notation

$$q[i,j] = s[i,j] + q_0[i,j] + q_1[i,j]2^{-1} + q_2[i,j]2^{-2} + \dots,$$

where s is the sign bit, q_0 is the MSB, q_1 is the next MSB, etc. The scale factor would then be coded in a header, along with such factors as the image dimensions, pixel depth, and color space. Two new symbols are used in the coder:

- *Zero-tree root (ZTR)*: first zero that flows down to all zero remaining tree at current bit plane.
- *Isolated zero (IZ)*: a zero at this position in current bit plane, but not further down the data tree.

Other messages used in the EZW coder are $+$ and $-$ to indicate the sign of a subband value (wavelet coefficient) and then 0 and 1 to represent the binary values of the lower significance bits.

We proceed iteratively, with two passes through the bit-plane data corresponding to Figure 9.6–9. The first pass is called the *dominant pass* and creates a list of those coefficients not previously found to be dominant. The second *subordinate pass* refines those coefficients found to be significant in a previous dominant pass. The dominant pass scanning order is zigzag, right-to-left and then top-to-bottom, within each scale (resolution), before proceeding to the next higher scale. On subsequent dominant passes, only those coefficients not yet found to be significant are scanned.

EZW Algorithm

A simplified version of the algorithm can be written as follows:

1. Set threshold T_1 such that $\{2T_1 > \max |c_{i,j}| \text{ and } T_1 \text{ is smallest such}\}$, and normalize the data.
2. Set $k = 0$.
3. Conduct dominant pass by scanning through the data $q_k(i,j)$ and for each newly significant pixel, output a message: $+$, $-$, IZ , and ZTR for *conditional adaptive arithmetic coder* (CAAC).⁷ But first time through, the coder must visit all the samples.
4. Conduct a subordinate pass by scanning through the data $q_k(i,j)$ to refine pixels already known to be significant in the current bit plane. Such bit values may be output directly or also given to CAAC.
5. Set $k \leftarrow k + 1$.
6. Stop (if the stopping criterion is met) or go to step 3.

The stopping criterion may be a fixed number of bit planes coded, but is usually a given numerical level. This is where the minimum step size of the quantizer comes in. If we want to code at highest quality with a step size of Δ for each subband, then we will stop at $k = \arg\{T_k = \Delta\}$. If we decode the full bitstream, we get this quality. If we only decode the earlier part of the bitstream, we get a lesser quality, i.e.,

⁷In the literature, the abbreviation CABAC is widely used. It stands for conditional adaptive binary arithmetic coder. The acronym CAAC is not widely used.

corresponding to the quantizer step of whatever bit plane we stop at. Such a coder is thus fully bit-plane embedded and permits fine-grain scalability in quality (PSNR) or alternatively in bitrate. To achieve a fixed bitrate, one simply stops decoding when that number of bits is reached.

We note that a scalable image coder really has three parts. First is the *precoder*, which codes the data into a coded image file archive with only a maximum bitrate or quality in mind. The second part is the *extractor*, which pulls bits from the precoder's archive. These two steps together make up the scalable encoder. The last part is the decoder.

Set Partitioning in Hierarchical Trees Coder

The famous set partitioning in hierarchical trees (SPIHT) coder of Said and Pearlman [17] replaces the zero trees with three lists that indicate significance and insignificance of sets of pixels and their descendants. Its generally higher performance compared to EZW has made it the coder of choice for many applications, and it is currently the one to beat in the research literature. The parent-child relation in SPIHT is somewhat different from EZW, as shown in Figure 9.6–10. In the coarsest subband LL_N , we have sets of 2×2 pixels that each initiate three trees, in horizontal, diagonal, and vertical directions. Pixels in locations $(2i, 2j)$ do not have descendants.

SPIHT introduces three lists of subband pixels (wavelet coefficients):

1. List of insignificant pixels (LIP)
2. List of significant pixels (LSP)
3. List of insignificant sets (LIS)

The LIS is further broken down into two types of sets of insignificant pixels: *type A*, in which all descendants are zero, and *type B*, in which all grandchildren and further descendants are zero.

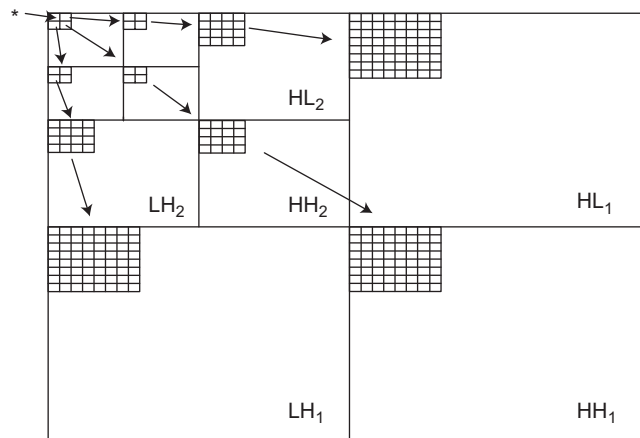


FIGURE 9.6–10

Illustration of SPIHT parent-child dependencies for a three-level tree.

To describe the algorithm more easily, we define new sets with respect to a pixel at location $[i,j]$:

- $\mathcal{C}[i,j]$ denotes its *children*.
- $\mathcal{D}[i,j]$ denotes its *descendants*.
- $\mathcal{G}[i,j]$ denotes its *grandchildren*.

Clearly, we have $\mathcal{G} = \mathcal{D} - \mathcal{C}$, where the sign $-$ denotes set subtraction. Also, for a given set of pixels \mathcal{B} , we define the indicator function

$$S_k(\mathcal{B}) = \begin{cases} 1, & \mathcal{B} \text{ contains a pixel with } q_k[i,j] = 1, \\ 0, & \text{otherwise,} \end{cases}$$

where k is the bit plane index and 0 is the MSB.

We are now in a position to state the SPIHT coding algorithm.

SPIHT Coding Algorithm

1. Set $k = 0$, LSP = ϕ , LIP = {all coordinates $[i,j]$ of LL_N }, LIS = {all coordinates $[i,j]$ of LL_N that have children}.
2. *Significance pass*
 - For each $[i,j] \in \text{LIP}$, **output** $q_k[i,j]$. If $q_k = 1$, **output** sign bit $s[i,j]$ and move $[i,j]$ to end of LSP.
 - For each $[i,j] \in \text{LIS}$:
 - If set is of type A, **output** $S_k(\mathcal{D}[i,j])$. If $S_k(\mathcal{D}[i,j]) = 1$, then:
 - * For each $[l,m] \in \mathcal{C}[i,j]$, **output** $q_k[l,m]$. If $q_k[l,m] = 0$, add $[l,m]$ to the LIP. Else, **output** $s[l,m]$ and add $[l,m]$ to LSP.
 - * If $\mathcal{G}[i,j] \neq \phi$, move $[i,j]$ to end of LIS as set of type B. Else delete $[i,j]$ from LIS.
 - If set is of type B, **output** $S_k(\mathcal{G}[i,j])$. If $S_k = 1$, then add each $[l,m] \in \mathcal{C}[i,j]$ to the end of the LIS as sets of type A and delete $[i,j]$ from LIS.
3. *Refinement pass*
 - For each $[i,j] \in \text{LSP}$, **output** $q_k[i,j]$ using *old* LSP.
4. Set $k \leftarrow k + 1$ and go to step 2.

Note that for sets of type A, we process all descendants, while for sets of type B, only grandchildren are processed. The SPIHT decoder is almost identical with the encoder. Initialization is the same, then the data are input from the binary codestream at each point in the coding algorithm where an output is indicated. In this way the decoder is able to process the data in the same sequence as the coder, and so, no positional information has to be transmitted. While this is very efficient, one bit error somewhere in the SPIHT codestream can make the decoder lose track and start producing useless results. In particular, a bit error in the significance pass will cause such a problem. Of course, it can be said that any image coder that uses VLC potentially has this problem. The SPIHT coder outputs can also be arithmetic coded for added coding efficiency.

Embedded Zero Block Coder

The main idea in the embedded zero block coder (EZBC) is to replace the zero tree across the scales (subbands) with separate quadrees within each subband to indicate the significance of the data. This permits resolution scalability to accompany quality or SNR scalability in a more efficient manner than with zero trees, which necessarily carry information on all scales higher than the present one. We need some new notation to describe the quadrees.

We denote quantized subband pixels as $c(i,j)$ with MSB $m(i,j)$ and pixel value in subband k written as $c_k(i,j)$. Also note that as quadtree level l goes up, quadtree resolution goes down, with level $l = 0$ being full resolution.

Definition 9.6–1: EZBC Algorithm Notation

$QT_k[l](i,j) \triangleq$ quadtree data at quadtree level l , in subband k , at position (i,j) ,
 $K \triangleq$ number of subbands,
 $D_k \triangleq$ quadtree depth for subband k ,
 $D_{\max} \triangleq$ maximum quadtree depth,
 $LIN_k[l] \triangleq$ list of insignificant nodes for subband k and quadtree level l ,
 $LSP_k \triangleq$ list of significant pixels from subband k ,
 $S_n(i,j) \triangleq$ significance of node (i,j) at bit plane n ,

$$S_n(i,j) \triangleq \begin{cases} 1, & \text{if } n \leq m(i,j), \\ 0, & \text{else.} \end{cases}$$

Coder initialization:

$$QT_k[0](i,j) = |c_k(i,j)|$$

$$QT_k[l](i,j) = \max\{QT_k[l-1](2i, 2j), QT_k[l-1](2i-1, 2j), QT_k[l-1](2i, 2j-1), QT_k[l-1](2i-1, 2j-1)\}.$$

$$LIN_k[l] = \begin{cases} \{(0,0)\}, & l = D_k, \\ \phi, & \text{otherwise.} \end{cases}$$

EZBC Coding Algorithm

1. Initialization

- $LIN_k[l] = \begin{cases} \{(0,0)\}, & l = D_k, \\ \phi, & \text{otherwise,} \end{cases}$
- $LSP_k = \phi, \forall k$,
- $n = \lfloor \log_2 \max_{k,i,j} \{|c_k(i,j)|\} \rfloor$. (Note that here the bit-plane index is n and that it decreases with successive coding passes.)

2. For $l = 0, D_{\max}$:
 - For $k = 0, K - 1$
 - CodeLIN(k, l)
3. For $k = 0, K - 1$
 - – CodeLSP _{k}
 - $n \leftarrow n - 1$ and go back to step 2. Stop when hit target bits.

We now describe the functions CodeLIN(k, l) and CodeLSP _{k} .

CodeLIN(k, l)

- For each $(i, j) \in \text{LIN}_k[l]$:
 - **code** $S_n(i, j)$.
 - If $(S_n(i, j) = 0)$, (i, j) remains in $\text{LIN}_k[l]$.
 - Else:
 - * If $(l = 0)$, then **code** the sign bit of $c_k(i, j)$ and add (i, j) to LSP _{k} .
 - * Else CodeDescendentNodes(k, l, i, j).

CodeDescendentNodes(k, l, i, j)

- For each node $(x, y) \in \{(2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)\}$ of quadtree level $l - 1$ in subband k :
 - **code** $S_n(x, y)$.
 - If $(S_n(x, y) = 0)$, add (x, y) to LSP _{k} .
 - Else CodeDescendentNodes($k - 1, l, i, j$).

CodeLSP _{k}

- For each pixel $(i, j) \in \text{LSP}_k$, **code** bit n of $|c_k(i, j)|$.

The word **code** in this algorithm is like the word **output** in the SPIHT algorithm, in that they both specify algorithm output. Here, though, we take the algorithm output and send it to a CAAC rather than outputting it directly to the channel.

Context-Based AC in EZBC

The context-based AC used in EZBC is described next with reference to Figure 9.6–11. The context model is binary and is based on the quadtree neighbors of the current datum. There are eight nearest neighbors from the current subband and quadtree levels, plus one context node from the next higher subband level and next lower quadtree level, which has the same position. The actual value of this 9-bit context is determined by the values encountered at the previous location, as scanned in a normal progressive raster. In this way, dependence between bit planes, and with nearby prior encoded neighbors, is not lost and so efficient coding therefore results. Instead of using 2^9 states, states with similar conditional statistics are judiciously merged similarly to the method used in EBCOT and JPEG 2000. Keeping the number

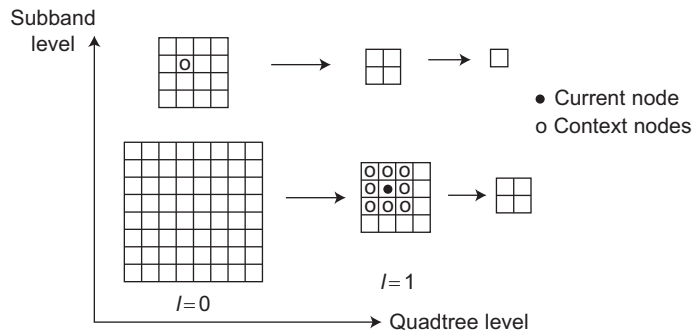
**FIGURE 9.6–11**

Illustration of context modeling for AC in EZBC.

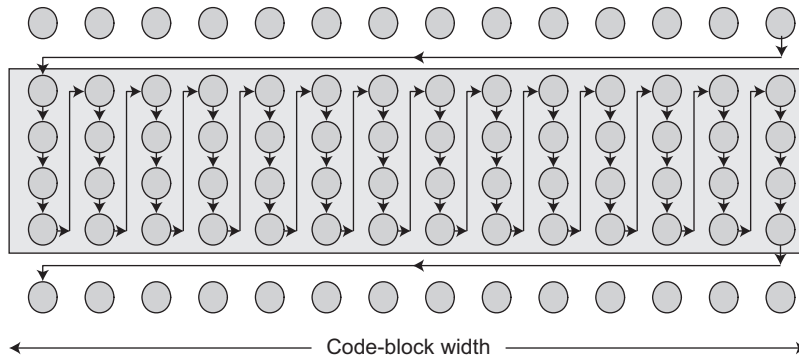
of these states small permits more rapid and accurate adaptation to varying statistics and hence better compression.

9.7 JPEG 2000

The JPEG 2000 standard is based on *embedded block coding with optimal truncation* (EBCOT) [19] and is similar to the other embedded algorithms already discussed, with the main exception that the subband values are separated into *code blocks* of typical size 64×64 , which are coded separately. Individual rate-distortion functions are established for each code block, and the overall total bits are apportioned to the code blocks based on the equal slope condition (9.6–3), thereby funneling additional bits to wherever they are most needed to maximize the decrease in the total distortion, assumed to be approximately additive over the code blocks. So, some optimality is lost by breaking up the code blocks, but increased performance is obtained by the functionality of being able to send bits to code blocks where they are most needed.⁸ The “optimal truncation” part of the EBCOT acronym refers to this bit assignment, since the code blocks’ bitstreams are embedded, and thus can be truncated to get reduced quality points. It also follows that if each code block is bit-plane coded down to a sufficiently low bit plane, then the optimization can be done after the bit-plane coding in a “postcompression” pass. Then only occasionally and at very high total bit values, would the binary version of the code block not be of sufficient accuracy.

EBCOT does not use any quadtree coding of significance at each bit-plane level. Rather each bit plane is coded in three so-called *fractional* passes. With reference to Figure 9.7–1, wavelet coefficients (subband samples) are scanned in a horizontal

⁸The relatively small code blocks permit a region of interest (ROI) type of scalability, wherein higher accuracy representation may be given to certain high-interest areas of the image.

**FIGURE 9.7–1**

An illustration of how JPEG 2000 coder scans a four-row stripe in a code block of wavelet coefficients (subband samples).

stripe fashion with a stripe width of 4 pixels vertically. A significance state is kept for each coefficient, and the first fractional pass *significant propagation* visits just those coefficients that neighbor a significant coefficient, since they have the highest likelihood of becoming significant and, when they do, will reduce the distortion the most for a given incremental number of bits. When a significant value is detected, its sign is coded again with a context-based arithmetic coder. A second fractional pass *magnitude refinement* codes those coefficients already found to be significant in a previous bit plane. The last *cleanup pass* treats those coefficients that have not yet been coded, as they have the least expected contribution to average distortion reduction. Here, a special *run mode* is used to efficiently deal with the expected runs of zero bit-plane values.

In order to make the the EBCOT method practical, there must be relatively efficient means to approximate the rate-distortion functions of the code blocks, and also to do the optimized truncation. Means of doing both are included in [8], which also treats certain simplificational differences between EBCOT and the JPEG 2000 standard.

9.8 COLOR IMAGE CODING

As we have seen in Chapter 6, color images are simply multicomponent vector images, commonly composed of red, green, and blue components. Normally images are stored in the transformed $Y C_r C_b$ color space, where often the two chroma channels C_r and C_b have been subsampled. Common subsampling strategies are labeled 4:2:2 and 4:2:0 and are illustrated in Figure 9.8–1, with 4:2:2 referring to the horizontal sampling rate of the Y , C_r , and C_b components, respectively, and meaning that

there are only two C_r and C_b samples for each four luma Y samples, i.e., the chroma C_r and C_b signals are decimated horizontally by the factor 2.

The commonly used notation 4:2:0 extends the chroma decimation into the vertical direction, again by the factor 2, meaning chroma components are decimated 2×2 . The justification for this decimation is that, as we have seen in Chapter 6, the HVS is less sensitive to spatial high frequencies in color than in luma. Sometimes, but not always, the chroma channels have been prefiltered to avoid aliasing error before being converted to 4:2:2 or 4:2:0. For definiteness, the full color space, with no chroma subsampling, is denoted 4:4:4.

In Figure 9.8–1, we have shown the chroma subsamples sited at the locations of a corresponding luma sample, but this is not commonly the case. Figure 9.8–2(b) shows an example of 4:2:0 with the chroma samples situated more symmetrically between luma samples and is used in the DV and HDV digital video tape standards, as well as in the DVD and Blu-ray video disks. Note that conversion of video with the color format of Figure 9.8–2(a) to that of Figure 9.8–2(b) can be effected by a unity gain chroma filter approximating a half-sample delay in each dimension.

Normally, each of the three components Y , C_r , and C_b are separately coded by any of the methods of this chapter, just as though they were separate gray-level or monochrome images. The new issue, thus, is to apportion the total bitrate to the three color components. Due to the color subsampling that is almost universal, the number of luma samples is twice that of each chroma component in 4:2:2 and four times in 4:2:0. Usually the chroma components are easier to code than the luminance, because of their general lack of detail and often lowpass character. Efforts have been made to come up with a total distortion criterion for the three components [20]. A general

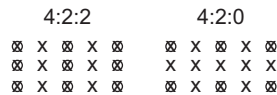


FIGURE 9.8–1

Illustration of color subsampling strategies 4:2:2 and 4:2:0.

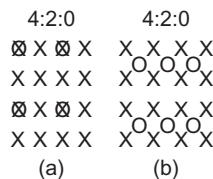


FIGURE 9.8–2

An illustration of different possible sitings of chroma subsamples in 4:2:0.

Table 9.8–1 PSNR for Some JPEG 2000 Test Set Images at 0.55 bpp

PSNR (dB)	<i>Lena</i>	<i>Barbara</i>	<i>Goldhill</i>	<i>Cafe</i>	<i>Bike</i>	<i>Hotel</i>
SPIHT	37.2	31.4	33.1	26.5	33.0	33.6
EZBC	37.6	32.2	33.6	27.1	33.7	34.7
JP2K	37.2	32.1	33.2	26.7	33.4	34.0
EBCOT	37.3	32.3	33.2	26.9	33.5	34.1
SPECK	37.1	31.5	33.0	26.3	32.7	—

good practice, however, is to assign bits to the components so as to make the quantizer step sizes about equal in the three color components, with this then generating the most pleasing visual result, and also somewhat higher PSNR for the two chroma components as compared to that of the luma channel.

Scalable Coder Results Comparison

Table 9.8–1, taken from [21], lists PSNR comparisons between some of the SWT coders, using a wide range of natural images that were part of the test set of the JPEG 2000 competition. If there is more than one version, we compare the scalable version. While the PSNR performance of all the coders are comparable, the advantage goes to the EZBC coder in this comparison. This is somewhat surprising in that the EBCOT and JPEG2000 (JP2K) coders are rate-distortion optimized. However, the advantage of the rate-distortion optimization is said to show up mainly in non-natural or composite images composed of combined images, text, and graphics [8]. Of course, in that case, it may be more optimal still to decompose such a complex composite image into its constituents and then code the graphics and the text separately.

9.9 DIRECTIONAL TRANSFORMS

New directional transforms have been developed for image coding. In one approach [22], an image block to be transform coded is first scanned in one of eight directions and then subject to 1-D DCT. After this step, 1-D DFTs are conducted on the rows or columns to complete the directional DCT transform. In addition to DC, horizontal, and vertical modes, the six other directions are shown in Figure 9.9–1, from [22]. Depending on the chosen mode, we see that the 1-D DCTs are of quite variable length, ranging from 1 to 8 and then down to 1 again for mode 3 on the 8×8 blocks shown.

Horizontal 1-D DCTs are used as the second stage transform in the top three modes, but vertical 1-D DCTs are used in the bottom three modes. Coded results demonstrate both objective (PSNRs) and subjective merit to this approach. A refined and extended version of this method, termed direction-adaptive partitioned block transform, is presented in [23] and shows considerable improvement over the earlier JPEG standard (cf. Section 9.5).

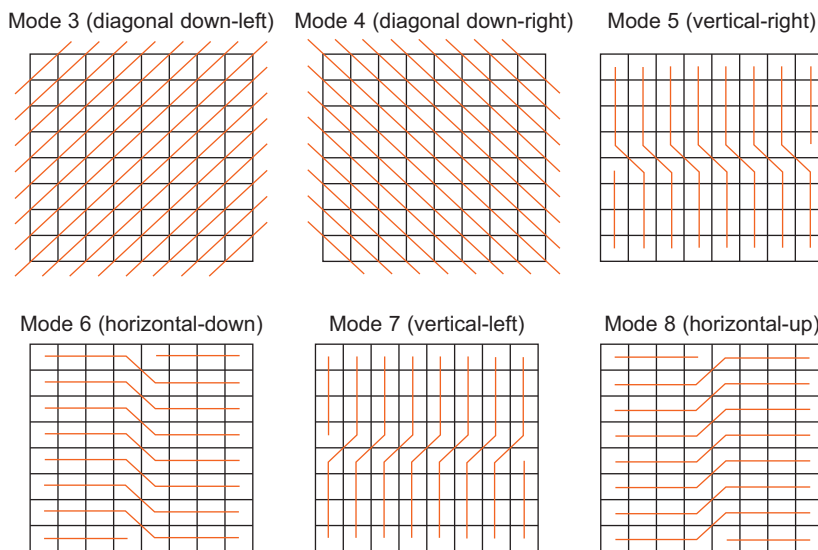
**FIGURE 9.9-1**

Illustration of six new modes for directional DCT in [22]. (© 2008 IEEE)

In an alternative approach, the SWT is used in a lifting implementation to perform directional filtering in a subband/wavelet image coder [24]. First, the local correlation is estimated, then the prediction operator in a 1-D lifting-based SWT is warped to this direction. Specifically, they modify the lifting implementation (see problem 14 in Chapter 5) of a vertical 1-D SWT by predicting the polyphase samples on odd lines from the neighboring polyphase samples on even lines with angles θ_i , $-4 \leq i \leq +4$, using sinc function interpolation to quarter-pixel accuracy. Then they do the same kind of modification to the horizontal 1-D SWT to provide their *adaptive directional lifting* (ADL) transform. Fixed ADL transforms are used in variable-size quadtree blocks with decomposition determined by BFOS search [14] using a Lagrangian cost function. An indication of the directions and segmentation is shown in Figure 9.9-2 from [24]. The ADL transform method was implemented into an EBCOT coder and compared to the JPEG 2000 coder on several test images. Figure 9.9-3a and Figures 9.9-3b show the comparison results for coding the Barbara image at 0.3 bpp with their ADL version of the 5/3 SWT, with the JP2K result in Figures 9.9-3a and ADL result in Figure 9.9-3b.

The PSNR comparison here is approximately 28 dB for the JPEG 2000 coder using the 5/3 SWT and about 30 dB for the ADL coder. Since the ADL transform is generally aligned with image structure, ringing artifacts are reduced and coding efficiency is improved versus the conventional 2-D separable SWT, for which Ding et al. [24] introduce the term *rectilinear* since the ADL is separable, too, just not

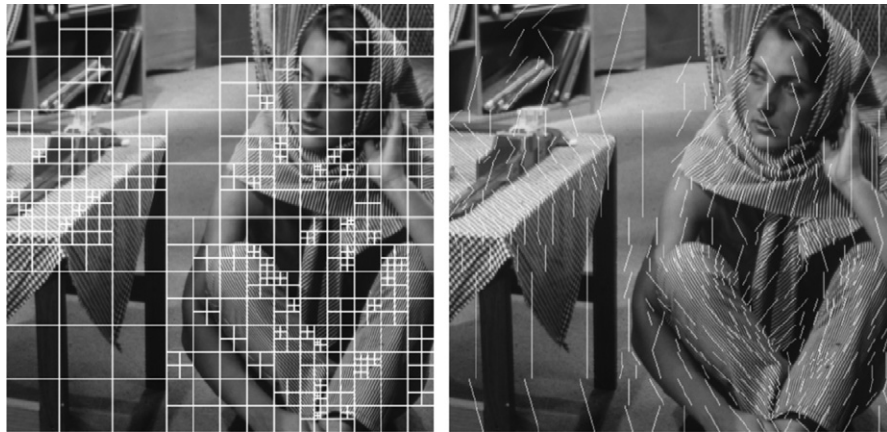
**FIGURE 9.9-2**

Illustration of ADL quadtree and chosen directions overlaid on the Barbara image. (from Ding et al. [24] © 2007 IEEE)

**FIGURE 9.9-3a**

Barbara JPEG 2000 coded with 5/3 SWT at 0.3 bpp. (from Ding et al. [24] © 2007 IEEE)

separable in strict horizontal and vertical terms. An ADL approach using directional multidimensional filters is presented in [25] and shows good performance with reduced complexity. A review article on both these approaches to directional transforms appears in [26].

**FIGURE 9.9–3b**

Barbara ADL coded with 5/3 SWT at 0.3 bpp. (from Ding et al. [24] © 2007 IEEE)

9.10 ROBUSTNESS CONSIDERATIONS

Almost all the specific image coding algorithms that we have discussed thus far are not robust to errors. This is because they use VLC. The Huffman code tree is complete, in the sense that all binary strings are possible. So a decoder, upon encountering a bit error, will start decoding erroneous values from that point forward, loosely referred to as *decoder failure*. Practical coding algorithms additionally have markers such as *end of block* (EOB) and other sync words, which serve to terminate or at least control the error propagation. These sync words are inserted into the coder bitstream at appropriate points. Also, alternative non-Huffman VLCs may be employed, whose code tree, while reasonably efficient, is not complete. Then illegal codewords can be detected and used to reset the decoder.

In practical coders, positional information and the block length in pixels are included in headers that start each code block. Also an overall header precedes the entire bitstream, giving image size, bit depth, color space, etc. Finally, the decoder needs to know how to act when it encounters a problem, e.g., if the code block is of fixed size, and the EOB comes too soon, indicating an error situation. A common response would be to discard the data in the current block, insert black into the now-missing block, and search for the next sync word, and start decoding again at that point. In essence, it is the *syntax* or structure of a coder, a structure that is created by the insertion of header and sync word information, that enables the decoder to detect errors. Then, if the decoder knows how to react to these errors, it can be considered at least somewhat robust. Of course, if errors are expected

to occur with some frequency, then error control coding and acknowledgement-retransmission (ACK/NACK) schemes can be used. Also, error concealment schemes can be employed at the decoder. More on this in the context of video coding will be presented in Chapter 13.

CONCLUSIONS

Image source compression is important for the transmission and storage of digital image and other 2-D data. Compression can be lossless, consisting of just an invertible transformation and variable-length coding, or lossy, employing a quantizer. Most practical applications to natural images involve lossy coding due to the significantly higher compression ratios that can be achieved. Lossless coding is generally confined to medical images, works of art, and other high-value images. Its compression ratios are very modest by comparison, typically around two to one or so. In between these two, there is the category *visually lossless* that means the HVS cannot detect the loss (in most cases); i.e., the loss is below the JND, and used in digital cinema compression.

The old standard image compression algorithm JPEG uses the DCT and is non-scalable. The more recent international standard *JPEG 2000* features an SWT and embedded scalable VLC. There is also the option for lossless coding in this coding algorithm. We briefly covered some new image techniques based on adaptive directional transforms, both DCT and SWT.

Lastly, we introduced the issue of coder robustness, achieved only by leaving in or reinserting some controlled redundancy that allows the decoder to detect and recover from even isolated single bit errors.

PROBLEMS

1. Interpret 8×8 block DCT image coding as a type of subband/wavelet coder with the DCT basis functions acting as the filters. What is the number of subbands? What is the 2-D decimation ratio? Do the filter supports of each subband overlap on the plane after subsampling? What is the passband of the filter used to produce the lowpass subband—i.e., the one including frequency $(0, 0)$? What is its minimum stopband attenuation?
2. In an optimal MSE scalar quantizer, the decision values must satisfy

$$d_i = \frac{1}{2}(r_i + r_{i+1}),$$

in terms of the neighboring representation values. This was derived as a stationary point, using partial derivatives, but here you are to conclude that it must be so, based on first principles. Hint: Consider the effect of moving the decision point d_i to the right or left. What does this do to the quantizer output?

3. An approximation had been derived (Panter and Dite, 1949) for so-called *fine scalar quantization* wherein the pdf of the signal x is approximately constant over a quantization bin, i.e.,

$$f_x(x) \approx f_x \left(\frac{1}{2} (d_{i-1} + d_i) \right), \text{ for } d_{i-1} < x \leq d_i, \text{ with } d_{i-1} \approx d_i,$$

resulting in the fine quantization approximation formula

$$D = \frac{1}{12L^2} \left[\int_{-\infty}^{+\infty} f_x(x)^{1/3} dx \right]^3.$$

- (a) Evaluate this approximate distortion in the case where x is $N(0, \sigma^2)$.
 (b) Evaluate in the case where x is Laplacian distributed,

$$f_x(x) = \frac{1}{\alpha} \exp -\frac{2}{\alpha} |x|, \quad \text{for } -\infty < x < +\infty, \text{ with } \alpha > 0,$$

and where $\alpha = \sqrt{2}\sigma$.

4. Work out the logarithmic bit assignment rule of (9.5–2) and (9.5–3). What are its shortcomings?
 5. This problem concerns optimal bit assignment across quantizers after a unitary transform, i.e., the optimal bit allocation problem. We assume

$$R = \sum r_n \quad \text{and} \\ D = \sum d_n,$$

where n runs from 1 to N , the number of coefficients (channels). Here, r_n and d_n are the corresponding rate and distortion pair for channel n . Assume the allowed bit allocations are M in number, $0 \leq b_1 < b_2 < \dots < b_M$, and that the component distortion-rate functions $d_n = d_n(b_m)$ are given for all n and for all m and is assumed to be convex.

- (a) Argue that the assignment $r_m = b_1$, for all m , must be on the optimal assignment curve as the lowest bitrate point, at total rate $R = Nb_1$.
 (b) Construct a straight line to all lower distortion solutions, and argue that the choice resulting in the lowest such line must also be on the optimal distortion-rate curve.
 (c) In part (b), does it suffice to try switching in the next higher bit assignment b_2 for each channel, one at a time?
 6. Making use of the SQ model

$$D = g\sigma^2 2^{-2R},$$

find the approximate MSE for DPCM source coding of a random field satisfying the AR model

$$x(n_1, n_2) = 0.8x(n_1 - 1, n_2) + 0.7x(n_1, n_2 - 1) - 0.56x(n_1 - 1, n_2 - 1) + w(n_1, n_2),$$

where w is a white noise of variance $\sigma_w^2 = 3$. You can assume that the coding is good enough so that $\hat{x}(n_1, n_2)$, the $(1,0)$ step prediction based on $\tilde{x}(n_1 - 1, n_2)$, is approximately the same as that based on $x(n_1 - 1, n_2)$ itself.

7. Repeat the VQ of [Example 9.3-3](#) with a different initial guess for the representation vectors: $\mathbf{r}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{r}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\mathbf{r}_3 = \begin{bmatrix} 1/2 \\ -1 \end{bmatrix}$. Compute the average least-squares error in each case, and determine which local minimum is better, the one found here or the one in [Example 9.3-3](#).
8. Consider image coding with 2-D DPCM, but put the quantizer *outside* the prediction loop (i.e., complete the prediction-error transformation ahead of the quantizer operation). Discuss the effect of this coder modification. If you design the quantizer for this *open-loop* DPCM coder to minimize the quantizing noise power for the prediction error, what will be the effect on the reconstructed signal at the decoder? (You may assume the quantization error is independent from pixel to pixel.) What should the open-loop decoder be?
9. Consider using the logarithmic bit assignment rule, for Gaussian variables,

$$B_i = \frac{B}{N} + \frac{1}{2} \log_2 \frac{\sigma_i^2}{\sigma_{\text{gm}}^2}, i = 1, \dots, N,$$

with $\sigma_{\text{gm}}^2 \triangleq (\prod_i \sigma_i^2)^{1/N}$ and N = number of channels (coefficients). Apply this rule to the following 2×2 DCT output variance set:

$$\text{coef. map} = \begin{bmatrix} 00 & 10 \\ 01 & 11 \end{bmatrix} \text{ and corresponding variances} = \begin{bmatrix} 22 & 4 \\ 8 & 2 \end{bmatrix}.$$

Assume the total number of bits to assign to these four pixels is $B = 16$. Resolve any possible negative bit allocations by removing that pixel from the set and reassigning bits to those remaining. Noninteger bit assignments are OK since we plan to use variable-length coding. Give the bits assigned to each coefficient and the total number of bits assigned to the four pixels.

10. In this problem, you are to use the software *VcDemo* [9] to perform both DCT and subband coding video compression on the Lena 512×512 monochrome image at a range of bitrates: 0.2–1.0. Then plot the MSE and PSNR results versus bitrate. Compare the results.
11. Reconcile the constant slope condition for bit assignment that we get from optimal channel (coefficient) rate-distortion theory in the continuous case, i.e., the constant slope condition $dD_i/dR_i = \lambda$, with the BFOS algorithm [14] for the

discrete case, where we prune the quantizer tree of the branch with minimal ratio of distortion increase divided by rate decrease, i.e., $\arg \min_i |\Delta D_i / \Delta R_i|$. Hint: Consider pruning a tree, as the number of rate values for each branch approaches infinity. Starting at a high rate, note that after a possible initial transient, we would expect to be repeatedly encountering all the channels (coefficients, values) at a near-constant slope.

12. Show that the distortion and rate models in (9.6–1) and (9.6–2) imply that in scalable SWT coding of two resolution levels, the lower resolution level must always be refined at the higher resolution level if

$$\sigma_{\text{wgm}}^2(\text{base}) > \sigma_{\text{wgm}}^2(\text{enhancement}),$$

where $\sigma_{\text{wgm}}^2(\text{base})$ is the weighted geometric mean of the subband variances at the lower resolution and $\sigma_{\text{wgm}}^2(\text{enhancement})$ is the geometric mean of those subband variances in the enhancement subbands.

APPENDIX ON INFORMATION THEORY

In order to deal with efficient transmission or storage of images, we need a measure of information to enable us to quantify our compression results. Clearly this is true in the transmission of information to a distant source over a communications channel, but less clearly the same situation is present when we store and later retrieve information in a memory via the so-called storage channel. Excellent texts on information theory are Gallagher [1] and Cover and Thomas [5].

Information Measure

Assume that a source has M messages to send to a remote location. We could say that the amount of information is M or we could use any monotonic increasing function of M to measure the information. Now consider two independent sources, the first one with M messages and the second one also with M messages. The total number of messages is now the square M^2 . On the other hand, one feels intuitively that the amount of information has only doubled here. If we adopt a logarithmic measure of information, then the amounts of information in the two cases are $\log_2 M$ and $2 \log_2 M$, respectively, consistent with a doubling of information.

In our seemingly digital world, we transmit bits and so may like to represent information in terms of bits. For M messages, we would need about $\log_2 M$ bits to represent them; for example, for $M = 8$, we represent each message with a symbol consisting of a binary string of $3 = \log_2 8$ bits. These binary symbols would be sent over a channel, and then the receiver would perform the inverse transformation back to the message set, such that receiving the binary string 010, the receiver would output the message 3. More generally, if two independent sources had different message numbers, say M_1 and M_2 , then their total or combined information would be, in terms of our logarithmic measure, $\log_2 M_1 + \log_2 M_2$,

Message	Symbol	Message	Symbol
1	000	5	100
2	001	6	101
3	010	7	110
4	011	8	111

and information would be an additive quantity expressed in bits. This is essentially the information concept of Hartley [27] that existed prior to Shannon's classic work [28], Claude Shannon being considered the father of information theory. We see some limitations in Hartley's approach; for example, what if the two sources are not independent? Certainly in the extreme case where two sources are totally dependent, the combined and single information should be the same. In between, we would expect a good measure of combined information to increase as the dependence between two sources decreases, reaching the additive (double) value in the limit of total independence of the sources.

There is another problem with Hartley's measure $\log_2 M$, in that there is no reason to believe that it is the smallest number of bits necessary for a general source with M messages. In the interests of compression and efficiency, we want a measure of the least amount of bits necessary to represent or store a message source. Now, considering again just one source with M messages, Hartley's information measure represents each message with the same number of bits $\log_2 M$. If all the messages are equally likely, this is reasonable, but what if they are not? Let's take the case of a source of M messages $\{m_1, m_2, \dots, m_M\}$ with probabilities p_1, p_2, \dots, p_M satisfying $p_i \geq 0$, for each i and $\sum_{i=1}^M p_i = 1$. Again, our intuition tells us that improbable messages carry more information than probable ones. One could say that it is no news when it snows in Alaska! So some kind of increasing function of $1/p_i$ may be a good measure of information. Interestingly, in the case where all M messages are equally likely, we have $1/p_i = M$, and thus we can match Hartley's information measure with $\log_2(1/p_i)$ bits for each message.

Definition 9.A-1: Source Entropy

The overall average information or *entropy* of source X then becomes

$$\begin{aligned}
 H(X) &\triangleq \sum_{i=1}^M p_i \log \frac{1}{p_i} \\
 &= E \left[\log \frac{1}{p(X)} \right].
 \end{aligned}$$

The Shannon information measure *source entropy* was first presented in his classic paper [28], where he showed this entropy to be *the unique measure* satisfying some simple continuity and compound message properties like what we have just explored. We will take Shannon's entropy as our information measure in this course.

We usually choose 2 as our logarithm base (i.e., $\log = \log_2$), and we speak of *bits*. However, other choices are sometimes used. For example, for four-value logic,

it would be convenient to measure information using \log_4 , and we could speak of ‘quads.’ Sometimes for mathematical simplicity the logarithm base is chosen as e , giving natural logarithms (i.e., $\log = \ln$), and we speak of *nats*. Converting between two scales (e.g., nats and bits) is simple. Just use the relation $\log_2 x = \ln x / \ln 2$. Then H in bits equals H in nats divided by $\ln 2 = 0.69315$, or equivalently multiplied by $\log_2 e = 1.44267$. Note that H in bits is larger than H in nats (by about 44%) because the base 2 is smaller than the base e .

Example 9.A–1: Discrete Message Source

Again, we take the case of a source of M messages $\{m_1, m_2, \dots, m_M\}$ with probabilities p_1, p_2, \dots, p_M satisfying $p_i \geq 0$, for each i and $\sum_{i=1}^M p_i = 1$. In particular, we take $M = 4$ with probabilities as given in the following table:

p_1	p_2	p_3	p_4
1/2	1/4	1/8	1/8

Then the information per symbol becomes

$-\log p_1$	$-\log p_2$	$-\log p_3$	$-\log p_4$
1	2	3	3

and we can construct binary strings to represent the messages, called *source codewords*, using the tree structure in Figure 9.A–1.

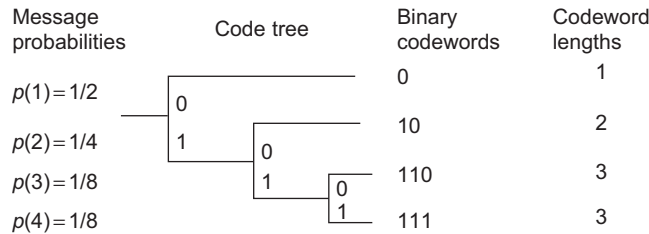


FIGURE 9.A–1

A convenient tree structure to calculate binary codewords given message probabilities.

Calculating the *average codeword length* \bar{l} , we find for this example

$$\begin{aligned}\bar{l} &= \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 \\ &= 1.75 \text{ bits}\end{aligned}$$

Note that the individual codeword lengths are equal to $-\log p_i$, and so the average codeword length is equal to the entropy here. In the general case, this is not so. It only occurs here because this source is dyadic; that is, all individual probabilities are of the form $p_i = 2^{-k}$ for some positive integer k .

We note that Shannon's entropy, here 1.75 bits, is less than Hartley's information measure of 2 bits and that this efficiency is achieved in this simple example by the variable-length binary codewords shown. Here are some properties of entropy.

Lemma 1 Entropy

For any source (random variable) X , we have $H(X) \geq 0$.

Proof Consider a probability p . We know $0 \leq p \leq 1$, and so $\log_2 p \geq 0$. Then $H(X) = E\left[\log \frac{1}{p}\right] \geq 0$.

Entropy is a measure of uncertainty. Consider a random variable X taking on M fixed values. If one of the values has probability 1, then all the remaining $M - 1$ values have probability zero, and $H = 1 \log_2 1 + 0 \log_2 \infty = 0$ since we take $0 \log_2 \infty$ at its limiting value $\lim_{k \rightarrow \infty} \frac{1}{k} \log_2 k = 0$. In fact, the maximum value of the entropy occurs when all the probabilities are equal (i.e., $p_i = \frac{1}{M}$ where $H = \sum_{i=1}^M \frac{1}{M} \log_2 M = \log_2 M$). We have the following theorem.

Theorem 9.A-1: Bounds on Entropy

For a random variable taking on M discrete values, we have the following bounds on entropy:

$$0 \leq H(X) \leq \log_2 M,$$

where the upper bound occurs when all values are equally likely and the minimum value 0 occurs when any one of the M values has probability 1.

Proof The upper limit follows since \log is a concave function (i.e., $\sum p_i \log \frac{1}{p_i} \leq \log \sum p_i \frac{1}{p_i} = \log M$). ■

Given two random variables X and Y , we can define joint and conditional entropy, respectively, as

$$H(X, Y) = \sum_{x,y} p(x, y) \log \frac{1}{p(x, y)} \quad \text{and} \quad H(Y|X) = \sum_{x,y} p(x, y) \log \frac{1}{p(y|x)}.$$

We then have the following theorem.

Theorem 9.A-2: Chain Rule for Entropy

For any two random variables X and Y , we have

$$H(X, Y) = H(X) + H(Y|X).$$

Proof Result is immediate upon substitution of the various definitions and recalling the chain rule for probabilities, we have $p(x, y) = p(x)p(y|x)$. ■

Now, when the random variables are independent, i.e., $p(y|x) = p(y)$, we have $H(X, Y) = H(X) + H(Y)$. Note that the chain rule can be extended to N random variables by repeated application, so we also have

$$H(X_1, X_2, \dots, X_N) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \dots,$$

and in the case where the X_i are jointly independent, this simplifies to $H(X_1, X_2, \dots, X_N) = H(X_1) + H(X_2) + H(X_3) + \dots$

A most important quantity in information theory is *mutual information* between random variables X and Y , defined as

$$I(X; Y) \triangleq \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

We speak of the mutual information *between* X and Y since it easily follows that $I(X; Y) = I(Y; X)$. Mutual information is related to entropy as

$$I(X; Y) = H(X) - H(X|Y), \quad (9.A-1)$$

which follows immediately from the definitions and the fact that $\frac{p(x,y)}{p(x)p(y)} = \frac{p(x|y)}{p(x)}$. In words, we say that the mutual information between X and Y is equal to the uncertainty (entropy) of X less the uncertainty that remains in X given that Y is known. Thus, if X is the input to a communications channel (link) and Y is its output, it seems that $I(X; Y)$ might actually measure the amount of information carried by Y about X . In fact, the channel coding theorem actually proves this and shows that the *channel capacity* (suitably defined) can be obtained by maximizing the mutual information.

Theorem 9.A-3: Mutual Information and Entropy

$$0 \leq I(X; Y) \leq H(X),$$

$$H(X|Y) \leq H(X).$$

Proof By (9.A-1), both of these two equations will be true if we can show mutual information $I(X; Y) \geq 0$, because entropy and hence conditional entropy are always nonnegative. To show that mutual information is always positive, we proceed as

$$\begin{aligned} I(X; Y) &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= (\log e) \sum_{x,y} p(x,y) \ln \frac{p(x,y)}{p(x)p(y)}. \end{aligned}$$

Now, since $\ln x \leq x - 1$ for $x > 0$, we can say that $\ln \frac{1}{x} \geq 1 - x$ there. Substituting x with $\frac{p(x)p(y)}{p(x,y)}$, we then obtain

$$\begin{aligned} \sum_{x,y} p(x,y) \ln \frac{p(x,y)}{p(x)p(y)} &\geq \sum_{x,y} p(x,y) \left[1 - \frac{p(x)p(y)}{p(x,y)} \right] \\ &= \sum_{x,y} p(x,y) - \sum_{x,y} p(x)p(y) \\ &= 1 - 1 \\ &= 0, \end{aligned}$$

as was to be shown. ■

Data Compression

We do not know yet that entropy is the minimum amount of information to represent a source, but a central result in information theory called the source coding theorem will show that it is. We start by looking at a variable-length code such as seen earlier in [Example 9.A–1](#) corresponding to a tree with a branching factor of 2. The messages to be coded are at the leaves of the tree, and the codeword bits are along the branches. This makes what is called a *prefix code*, one where no codeword is the prefix of another, and so guarantees unique decodability. We next show that codeword lengths must satisfy the Kraft inequality.

Theorem 9.A–4: Kraft Inequality

Let the M codewords of a binary prefix code have the lengths $l_1 \leq l_2 \leq \dots \leq l_M$; then these codeword lengths must satisfy the inequality

$$\sum_{i=1}^M 2^{-l_i} \leq 1.$$

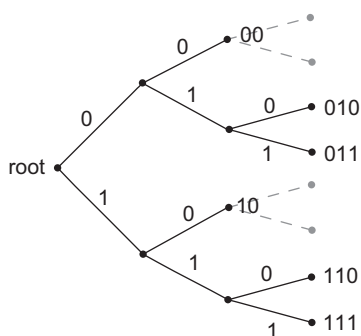
This is also sufficient for the existence of a code with these codeword lengths.

Proof Let the longest codeword have length $l_{\max} \triangleq l_M$; then a full or complete binary tree will have $2^{l_{\max}}$ leaves.

As can be seen from the example in [Figure 9.A–2](#), each codeword with length l_i is seen to prune off a total of $2^{l_{\max}-l_i}$ leaves. Taking the sum of these removed leaves, we have $\sum_{i=1}^M 2^{l_{\max}-l_i} \leq 2^{l_{\max}}$. Canceling out $2^{l_{\max}}$ from both sides, we have the desired result. Existence comes from the fact that, because of this inequality, there is always enough room for the next codeword to be assigned. ■

Now, the set of codewords has probabilities p_1, p_2, \dots, p_M , that sum to 1. Thus the Kraft inequality tempts us to write

$$p_i = 2^{-l_i},$$


FIGURE 9.A-2

An illustration of the variable-length code tree embedded in a full tree.

the only problem being that the corresponding codeword length $l_i = \log_2 \frac{1}{p_i}$ may not be an integer. However, if it is an integer, i.e., we have a dyadic source, then the average codeword length \bar{l} will be

$$\begin{aligned}\bar{l} &= \sum l_i p_i \\ &= \sum \left(\log_2 \frac{1}{p_i} \right) p_i \\ &= H(X).\end{aligned}$$

So, if we have such a dyadic source, by the Kraft inequality, there is a binary prefix code for that source such that the entropy $H(X)$ equals the average codeword length. But what about the general case when $\log_2 \frac{1}{p_i}$ is not an integer? In that case, we can round up this value to the next integer, calling the result l_i^* . This will make 2^{-l_i} smaller and thus ensure the existence of a prefix code. Then the average codeword length will be

$$\begin{aligned}\bar{l} &= \sum l_i^* p_i \\ &= \sum \left\lceil \log_2 \frac{1}{p_i} \right\rceil p_i, \quad \text{where } \lceil \cdot \rceil \text{ is the greatest integer function} \\ &< \sum \left(\log_2 \frac{1}{p_i} + 1 \right) p_i \\ &< H(X) + 1.\end{aligned}$$

We have thus established the following theorem.

Theorem 9.A-5: Source Coding

The average codeword length of an optimal binary prefix code satisfies

$$H(X) \leq \bar{l} < H(X) + 1.$$

This theorem essentially justifies the definition of entropy as a measure of the uncertainty, or what is the same the information content, of a discrete message source. Its proof suggests a way of getting a valid code—i.e., setting $l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$ —referred to as *Shannon coding*. It turns out that Huffman coding, to be introduced in Section 9.4, is usually somewhat better than Shannon coding and, in fact, can be shown to be optimal in the sense of minimizing the average codeword length [1, 5].

Instead of coding individual messages from the source (random variable) X , we could code vectors of length n source symbols X_n . The resulting vector source is called the extended source. Then applying the preceding result to the extended source, we get

$$H(X_n) \leq \bar{l}_n < H(X_n) + 1. \quad (9.A-2)$$

But if we assume that the source is i.i.d., then the individual entropies add and $H(X_n) = nH(X)$. Since we are coding n messages at a time, we have $\bar{l}_n = n\bar{l}$, so that upon insertion into (9.A-2), we get

$$H(X) \leq \bar{l} < H(X) + \frac{1}{n},$$

with the source entropy giving even a tighter bound on what can be achieved in terms of source-coding efficiency. In the limit as $n \rightarrow \infty$, we have $\bar{l}_\infty = H(X)$.

In summary, we have thus shown that entropy is the desired mathematical information (uncertainty) measure that can be used to characterize sources, including image sources, in terms of the minimum amount of bits to store or transmit them to a remote location. However, another way of looking at this is through a consequence of the law of large numbers.

Asymptotic Equipartition Principle

By the law of large numbers (LLN), if we have an i.i.d. sequence of random variables X_1, \dots, X_n , the sample average converges to the expected value $E[X_1] \triangleq E[X]$. The asymptotic equipartition principle (AEP) considers the information random variables $-\log p(X_i)$ and applies the LLN to their sample average

$$-\frac{1}{n} \sum_{i=1}^n \log p(X_i) = -\frac{1}{n} \log p(X_1, \dots, X_n).$$

Now the expected value of the random variable $-\log p(X_i)$ is

$$\begin{aligned} E[-\log p(X_1)] &= \sum_{i=1}^n p_i \log p_i \\ &= H(X_1) \triangleq H(X). \end{aligned}$$

Thus by the LLN, we have for large n that

$$p(X_1, X_2, \dots, X_n) = p(X_n) \approx 2^{-nH},$$

called the AEP [1, 5]. The set of such \mathbf{x}_n is called the *typical set* and denoted \mathcal{A}^n . Neglecting other probabilities than those \mathbf{x}_n in the typical set, we have

$$\begin{aligned} 1 &= \sum_{\text{all } \mathbf{x}_n} p(\mathbf{x}_n) \\ &\approx \sum_{\mathbf{x}_n \in \mathcal{A}^n} 2^{-nH} \\ &= |\mathcal{A}^n| 2^{-nH}, \end{aligned}$$

where $|\mathcal{A}^n|$ is defined as the number of vectors in the typical set \mathcal{A}^n . Solving for $|\mathcal{A}^n|$, we get that with high probability, the number of vectors we have to code is $|\mathcal{A}^n| = 2^{nH}$. Now, the number of fixed-length binary codewords of length L is 2^L , so we need length $L = nH$ fixed-length codewords to code the vector (n th extension) source, or equivalently an average of H bits per source message.

Continuous Sources

If a source X takes on a continuum of values, then the entropy as defined here can be infinite. In this case we introduce a new type of entropy, with an integral instead of a summation. The resulting quantity is called *differential entropy* and shares only some properties of entropy.

Definition 9.A–2: Differential Entropy

Let X be a continuous random variable (continuous source) with pdf $f(x)$, then we define differential entropy as

$$h(X) \triangleq \int_{\mathcal{S}} f(x) \log f(x) dx,$$

where \mathcal{S} is the support of f , i.e., $\mathcal{S} = \text{sup}\{f\} \triangleq \{x | f(x) > 0\}$.

Differential or continuous entropy is always nonnegative, but generally has no finite upper bound. If we quantize such a continuous source X with step size Δ , we would expect to often get a finite value for the entropy, and calling the quantized random variable X_Δ , it can be shown that

$$H(X_\Delta) + \log \Delta \approx h(X),$$

whenever Δ is sufficiently small. The argument essentially reduces to approximation of the above integral with a Riemann sum with bin width Δ . Turning this result around, we get an approximate result for the entropy of a quantized continuous random variable with quantizer sufficiently small step size of Δ ,

$$H(X_\Delta) \approx h(X) + \log \frac{1}{\Delta}, \quad (9.A-3)$$

relating differential entropy to entropy.

By our previous results for discrete random variables, $H(X_\Delta)$ is the number of bits to represent the quantized source X_Δ asymptotically in terms of sequences of samples of length n .

Example 9.A–2: (Gaussian Random Variable)

Let the random variable X be Gaussian distributed as $N(0, \sigma^2)$, then we have differential entropy

$$\begin{aligned} h(X) &= - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \left[-\frac{x^2}{2\sigma^2} - \ln\sqrt{2\pi\sigma^2}\right] dx \\ &= \frac{\sigma^2}{2\sigma^2} + \ln\sqrt{2\pi\sigma^2} \\ &= \frac{1}{2} + \frac{1}{2} \ln 2\pi\sigma^2 \\ &= \frac{1}{2} \ln 2\pi e\sigma^2 \quad \text{nats.} \end{aligned}$$

Converting to base 2 logarithms, we have $h(X) = \frac{1}{2} \log_2(2\pi e\sigma^2)$. We thus see that for a Gaussian random variable, the differential entropy $h(X)$ varies with standard deviation σ as $\log_2 \sigma$. Also, when σ/Δ is large, i.e., when the quantization is fine enough, then $H(X_\Delta) \approx \frac{1}{2} \ln 2\pi e(\sigma^2/\Delta^2)$ by (9.A–3). ■

Rate-Distortion Theory

Rate-distortion theory is a branch of information theory that deals with the source coding of continuous random sources. Since in general a continuous source has infinite entropy, some kind of approximation must be made to allow its storage or transmission with a finite number of bits. Rate-distortion theory tries to formulate the optimal trade-off of rate versus distortion, with some common distortion measures being MSE and mean-absolute error. The simple quantizer offers a way to trade off rate versus error. Generally, the MSE of a quantizer can be expressed in terms of the quantizer step size Δ with a formula such as $\text{MSE} = k\Delta^2$, which often holds for sufficiently small Δ , where k is a constant that depends on the pdf of the source. Adopting MSE as the distortion, we then have

$$D = k\Delta^2.$$

Using the result for a quantized Gaussian random variable in the preceding example, we can say that asymptotically for a uniform step-size quantizer, rate $R = H(X_\Delta)$ in bits per source symbol. Combining results, we get the operational rate-distortion

function of the quantizer, in this case as

$$\begin{aligned} R(D) &\approx \frac{1}{2} \log_2 [2\pi e(\sigma^2/\Delta^2)] \\ &= \frac{1}{2} \log_2 [k2\pi e(\sigma^2/D)]. \end{aligned}$$

However, this is only a suboptimal result for a Gaussian random variable. For one thing, it is only an approximation, and that for the case when $\Delta \ll \sigma$. Also, quantizers are not optimal in themselves. The optimal result has been derived using the rate-distortion theory, and it is stated next without proof.

Theorem 9.A-6: Gaussian Rate-Distortion Function

For a Gaussian source X distributed as $N(0, \sigma^2)$ and mean-square distortion measure $d(x, \hat{x}) = |x - \hat{x}|^2$, the (optimal) rate-distortion function is

$$R(D) = \begin{cases} \frac{1}{2} \log_2(\sigma^2/D), & 0 < D < \sigma^2, \\ 0, & D \geq \sigma^2. \end{cases}$$

A plot of the Gaussian rate-distortion function for $\sigma = 5$ is given in Figure 9.A-3. We see that, as D approaches zero, the rate R asymptotically becomes infinite for this continuous random variable, as it should. Also, the rate required above distortion

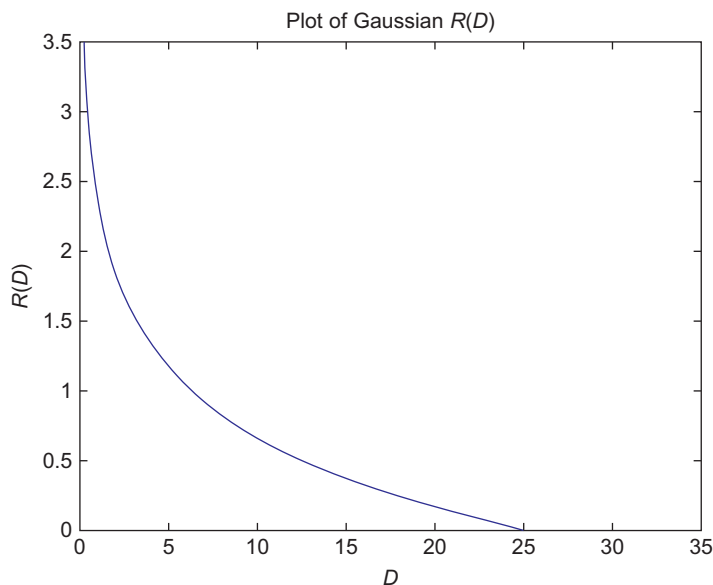


FIGURE 9.A-3

Plot of Gaussian $R(D)$ function for $\sigma = 5$.

$D = \sigma^2$ is zero, since the receiver is assumed to know the parameters of this distribution. So it can always estimate $\hat{X} = \mu (= 0 \text{ here})$ and incur a mean-square distortion equal to the variance σ^2 .

Another basic rate-distortion function for image coding is that of the Laplace random variable, which fits the measured density of image prediction errors and DCT coefficients other than DC. It has been found in closed form by Berger [29] for the case of mean-absolute distortion and is given here without proof.

Theorem 9.A-7: Laplace Rate-Distortion Function

For Laplace source of variance $\sigma^2 (> 0)$ and mean-absolute distortion measure $d(x, \hat{x}) = |x - \hat{x}|$, the (optimal) rate-distortion function is given in terms of $\alpha \triangleq \sqrt{2}/\sigma$:

$$R(D) = \begin{cases} -\log_2 \alpha D, & 0 < D < \alpha, \\ 0, & D \geq \alpha. \end{cases}$$

A plot of the Laplace rate-distortion function for $\sigma = 30$ is given in Figure 9.A-4. While the plot looks similar to that of the Gaussian, note that the standard deviations are quite different, as also is the distortion criterion, the Gaussian being square error while the Laplacian rate-distortion function is for absolute error. For a development of rate-distortion theory, see [1, 2, 5].

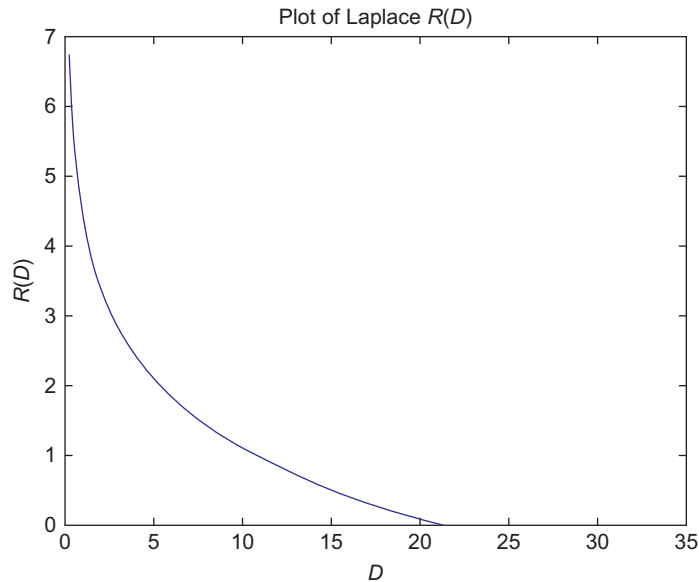


FIGURE 9.A-4

Plot of the Laplace rate-distortion function for $\sigma = 30$.

REFERENCES

- [1] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley, 1968.
- [2] R. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, 1992.
- [3] H. S. Malvar and D. H. Staelin, "The LOT: Transform Coding Without Blocking Effects," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 553–559, April 1989.
- [4] K. Ramchandran and M. Vetterli, "Best Wavelet Packet Bases in a Rate-Distortion Sense," *IEEE Trans. Image Process.*, vol. 2, pp. 160–175, April 1993.
- [5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd Ed., Wiley & Sons, Inc., New Jersey, 2006.
- [6] Z. Zhang and J. W. Woods, "Large Block VQ for Image Sequences," *Proc. IEE IPA-99*, Manchester, UK, July 1999.
- [7] H.-M. Hang and J. W. Woods, "Predictive Vector Quantization of Images," *IEEE Trans. Comm.*, vol. COM-33, pp. 1208–1219, November 1985.
- [8] D. S. Taubman and M. W. Marcellin, *JPEG 2000 Image Compression Fundamentals, Standards, and Practice*, Kluwer Academic Press, Norwell, MA, 2002.
- [9] R. L. Lagendijk, *VcDemo*. Available at <http://siplab.tudelft.nl/content/image-and-video-compression-learning-tool-vcdemo>, Delft University of Technology (DUT), Delft, The Netherlands.
- [10] K. Sayood, *Introduction to Data Compression*, Morgan Kaufman, 1996.
- [11] Y. H. Kim and J. W. Modestino, "Adaptive Entropy Coded Subband Coding of Images," *IEEE Trans. Image Process.*, vol. 1, pp. 31–48, January 1992.
- [12] S.-J. Choi, *Three-Dimensional Subband/Wavelet Coding of Video with Motion Compensation*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1996.
- [13] T. Naveen and J. W. Woods, "Subband Finite State Scalar Quantization," *IEEE Trans. Image Process.*, vol. 5, pp. 150–155, January 1996.
- [14] E. A. Riskin, "Optimal Bit Allocation via the Generalized BFOS Algorithm," *IEEE Trans. Inform. Theory*, vol. 37, pp. 400–402, March 1991.
- [15] W. Equitz and T. Cover, "Successive Refinement of Information," *IEEE Trans. Inform. Theory*, vol. 37, pp. 269–275, March 1991.
- [16] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Trans. Signal Process.*, vol. 41, pp. 3445–3462, December 1993.
- [17] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circ. and Sys. Video Technology*, vol. 6, pp. 243–250, June 1996.
- [18] S.-T. Hsiang and J. W. Woods, "Embedded Image Coding Using Zeroblocks of Subband/Wavelet Coefficients and Context Modeling," *Proc. ISCAS 2000*, May, Geneva, Switzerland, 2000.
- [19] D. S. Taubman, "High performance scalable image coding with EBCOT," *IEEE Trans. Image Process.*, vol. 9, pp. 1158–1170, July 2000.
- [20] P. H. Westerink, *Subband Coding of Images*, PhD thesis, Delft University of Technology, The Netherlands, October 1989.
- [21] S.-T. Hsiang, *Highly Scalable Subband/Wavelet Image and Video Coding*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, January 2002.
- [22] B. Zeng and J. Fu, "Directional Discrete Cosine Transforms—A New Framework for Image Coding," *IEEE Trans. Circ. and Sys. Video Technology*, vol. 18, no. 3, March, pp. 305–313, 2008.

- [23] C.-L. Chang, M. Makar, S. S. Tsai, and B. Girod, "Direction-Adaptive Partitioned Block Transform for Color Image Coding," *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1740–1755, July 2010.
- [24] W. Ding, F. Wu, X. Wu, S. Li, and H. Li, "Adaptive Directional Lifting-Based Wavelet Transform for Image Coding," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 416–427, February 2007.
- [25] Y. Tanaka, M. Ikehara, and T. Q. Nguyen, "Multiresolution Image Representation Using Combined 2-D and 1-D Directional Filter Banks," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 269–280, February 2009.
- [26] J. Xu, B. Zeng, and F. Wu, "An Overview of Directional Transforms in Image Coding," *Proc. of ISCAS*, pp. 3036–3039, 2010.
- [27] R. V. Hartley, "Transmission of Information," *Bell System Technical Journal*, vol. 7, pp. 535, 1928.
- [28] C. E. Shannon, "A Mathematical Theory of Communications," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.
- [29] T. Berger, *Rate Distortion Theory*, Prentice-Hall, Inc., Upper Saddle River, NJ 1971.

Three-Dimensional and Spatiotemporal Processing

10

In Chapter 2 we looked at 2-D signals and systems. Here, we look at the 3-D case, where often the application is to spatiotemporal processing, i.e., two spatial dimensions and one time dimension. Of course, 3-D more commonly refers to the three orthogonal spatial dimensions. Here, we will be mostly concerned with convolution and filtering, and so need a regular grid for our signal processing. While regular grids are commonplace in spatiotemporal 3-D, they are often missing in spatial 3-D applications. In any event, the theory developed in this section can apply to any 3-D signals on a regular grid. In many cases, when a regular grid is missing or not dense enough (often the case in video processing), interpolation may be used to infer values on a regular *supergrid*, maybe a refinement of an existing grid. Often such an interpolative resampling will be done anyway for display purposes.

10.1 3-D SIGNALS AND SYSTEMS

We start with a definition of a 3-D linear system.

Definition 10.1–1: 3-D Linear System

A system is linear when its operator T satisfies the equation

$$L\{a_1x_1(n_1, n_2, n_3) + a_2x_2(n_1, n_2, n_3)\} = a_1L\{x_1(n_1, n_2, n_3)\} + a_2L\{x_2(n_1, n_2, n_3)\},$$

for all (complex) scalars a_1 and a_2 , and for all signals x_1 and x_2 . When the system T is linear, we usually denote this operator as L .

Definition 10.1–2: 3-D Linear Shift Invariant System

Let the 3-D linear system L have output $y(n_1, n_2, n_3)$ when the input is $x(n_1, n_2, n_3)$; that is, $y(n_1, n_2, n_3) = L\{x(n_1, n_2, n_3)\}$. Then the system is 3-D LSI if it satisfies the equation

$$y(n_1 - k_1, n_2 - k_2, n_3 - k_3) = L\{x(n_1 - k_1, n_2 - k_2, n_3 - k_3)\},$$

for all integer shift values k_1, k_2 , and k_3 .

Often such systems are called linear *constant-parameter* or *constant-coefficient*. An example would be a multidimensional filter. In the 1-D temporal case, we generally need to specify initial conditions and/or final conditions. In the 2-D spatial case we often have some kind of boundary conditions. For 3-D systems, the solution space is a 3-D region, and we need to specify boundary conditions generally on all the boundary surfaces. This is particularly so in the 3-D spatial case. In the 3-D spatiotemporal case, where the third parameter is time, initial conditions often suffice in that dimension. However, we need both initial and boundary conditions to completely determine a filter output.

Definition 10.1–3: 3-D Convolution Representation

For an LSI system, with impulse response, h the input and output are related by the 3-D convolution,

$$\begin{aligned} y(n_1, n_2, n_3) &= \sum \sum \sum h(k_1, k_2, k_3) x(n_1 - k_1, n_2 - k_2, n_3 - k_3) \\ &= (h * x)(n_1, n_2, n_3), \end{aligned}$$

where the triple sums are over all values of k_1, k_2 , and k_3 , where $-\infty < k_1, k_2, k_3 < +\infty$.

We can see that 3-D convolution is a commutative operation, so that $h * x = x * h$ as usual.

Definition 10.1–4: 3-D Separability

A 3-D LSI system is a *separable system* if its impulse response factors or separates, such as $h(n_1, n_2, n_3) = h_1(n_1)h_2(n_2)h_3(n_3)$ or $h(n_1, n_2, n_3) = h_1(n_1, n_2)h_2(n_3)$. A signal is a *separable signal* if it separates into two or three factors. A *separable operator* factors into the concatenation of two or three operators, such as

$$T_{n_3}\{T_{n_2}\{T_{n_1}[\cdot]\}\} \quad \text{or} \quad T_{n_3}\{T_{n_1, n_2}[\cdot]\}.$$

Examples of separable operators in 3-D are the familiar transforms, i.e., Fourier, DFT, DCT, and SWT, which, in this way, extend to the 3-D case naturally.

Example 10.1–1: Separable System and Separable Convolution

A given LSI system has input x and output y and separable impulse response $h(n_1, n_2, n_3) = h_1(n_1, n_2)h_2(n_3)$. Expressing the 3-D convolution, we have

$$y(n_1, n_2, n_3) = \sum \sum \sum h_1(k_1, k_2)h_2(k_3)x(n_1 - k_1, n_2 - k_2, n_3 - k_3) \quad (10.1-1)$$

$$\begin{aligned}
&= \sum \sum h_1(k_1, k_2) \left[\sum_{k_3} h_2(k_3) x(n_1 - k_1, n_2 - k_2, n_3 - k_3) \right] \\
&= h_1(n_1, n_2) * [h_2(n_3) * x(n_1, n_2, n_3)] \\
&= h_2(n_3) * [h_1(n_1, n_2) * x(n_1, n_2, n_3)]^1
\end{aligned} \tag{10.1-2}$$

The result is a 2-D convolution with the 2-D signal consisting of x for each fixed value of n_3 , perhaps denoted as

$$x_{n_3}(n_1, n_2) \triangleq x(n_1, n_2, n_3),$$

followed by 1-D convolutions in n_3 for the 1-D output signal obtained by fixing n_1, n_2 and then ranging through all such n_1 and n_2 —i.e., $x_{n_1, n_2}(n_3) \triangleq x(n_1, n_2, n_3)$. Of course, and by linearity, the 1-D and 2-D convolutions in (10.1-1) can be done in either order. ■

Definition 10.1-5: 3-D Fourier Transform

$$X(\omega_1, \omega_2, \omega_3) \triangleq \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} \sum_{n_3=-\infty}^{+\infty} x(n_1, n_2, n_3) \exp -j(\omega_1 n_1 + \omega_2 n_2 + \omega_3 n_3).$$

Note that X is continuous and triply periodic with period $2\pi \times 2\pi \times 2\pi$.

Definition 10.1-6: Inverse 3-D Fourier Transform

$$x(n_1, n_2, n_3) = \frac{1}{(2\pi)^3} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} X(\omega_1, \omega_2, \omega_3) \exp +j(\omega_1 n_1 + \omega_2 n_2 + \omega_3 n_3) d\omega_1 d\omega_2 d\omega_3,$$

which amounts to a 3-D Fourier series, with the roles of “transform” and “signal” reversed, analogously to the 2-D and 1-D cases. ■

Properties of 3-D Fourier Transform

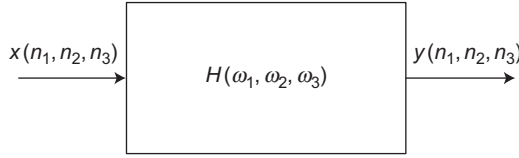
Ideal shift property (shift by k_1, k_2, k_3):

$$FT\{x(n_1 - k_1, n_2 - k_2, n_3 - k_3)\} = X(\omega_1, \omega_2, \omega_3) \exp -j(\omega_1 k_1 + \omega_2 k_2 + \omega_3 k_3)$$

Linearity:

$$FT\{a_1 x_1(n_1, n_2, n_3) + a_2 x_2(n_1, n_2, n_3)\} = a_1 X_1(\omega_1, \omega_2, \omega_3) + a_2 X_2(\omega_1, \omega_2, \omega_3)$$

¹Beware the notation here! The first convolution is just on the n_3 parameter. The second one is just on n_1, n_2 .

**FIGURE 10.1–1**

A 3-D system characterized by its frequency response.

Convolution:

$$FT\{x_1(n_1, n_2, n_3) * x_2(n_1, n_2, n_3)\} = X_1(\omega_1, \omega_2, \omega_3)X_2(\omega_1, \omega_2, \omega_3)$$

Separable operator:

$$FT\{h_1(n_1, n_2) * [h_2(n_3) * x(n_1, n_2, n_3)]\} = H_1(\omega_1, \omega_2)H_2(\omega_3)X(\omega_1, \omega_2, \omega_3)$$

This last property can be interpreted as a 2-D filtering cascaded with a 1-D filtering along the remaining axis. While this is not general, it can be expedient and sometimes is all that is required to implement a given transformation.

3-D Filters

Consider a constant coefficient 3-D difference equation

$$\begin{aligned} y(n_1, n_2, n_3) = & - \sum_{(k_1, k_2, k_3) \in \mathcal{R}_a - (0,0,0)} a_{k_1, k_2, k_3} y(n_1 - k_1, n_2 - k_2, n_3 - k_3) \\ & + \sum_{(k_1, k_2, k_3) \in \mathcal{R}_b} b_{k_1, k_2, k_3} x(n_1 - k_1, n_2 - k_2, n_3 - k_3), \end{aligned}$$

where \mathcal{R}_a and \mathcal{R}_b denote the 3-D filter coefficient support regions. By using linear- and shift-invariant properties of 3-D FTs, we can transform the preceding relation to multiplication in the frequency domain with a system function $H(\omega_1, \omega_2, \omega_3)$,

$$H(\omega_1, \omega_2, \omega_3) = \frac{B(\omega_1, \omega_2, \omega_3)}{A(\omega_1, \omega_2, \omega_3)},$$

given in terms of the Fourier transform $B(\omega_1, \omega_2, \omega_3)$ of the filter feed-forward coefficients $b(k_1, k_2, k_3) \triangleq b_{k_1, k_2, k_3}$ and the Fourier transform $A(\omega_1, \omega_2, \omega_3)$ of the filter feedback coefficients $a(k_1, k_2, k_3) \triangleq a_{k_1, k_2, k_3}$, where $a(0, 0, 0) = +1$. A 3-D LSI system diagram is shown in Figure 10.1–1.

10.2 3-D SAMPLING AND RECONSTRUCTION

The 3-D sampling theorem follows easily from the corresponding 2-D theorem in Chapter 2. The method of proof also extends, so that the same method can

be used. As before, we use the variables t_i for the continuous parameters of the function $x_c(t_1, t_2, t_3)$ to be sampled, with its corresponding continuous-parameter FT $X_c(\Omega_1, \Omega_2, \Omega_3)$.

Theorem 10.2–1: 3-D Sampling Theorem

Let $x(n_1, n_2, n_3) \triangleq x_c(n_1 T_1, n_2 T_2, n_3 T_3)$, with sampling periods, T_i , $i = 1, 2, 3$; then:

$$X(\omega_1, \omega_2, \omega_3) = \frac{1}{T_1 T_2 T_3} \sum_{k_1, k_2, k_3} X_c \left(\frac{\omega_1 - 2\pi k_1}{T_1}, \frac{\omega_2 - 2\pi k_2}{T_2}, \frac{\omega_3 - 2\pi k_3}{T_3} \right).$$

Aliasing occurs when the shifted FTs on the right overlap. This will occur when the T_i 's are not small enough. If X_c is ideally bandlimited in time and space, then $\frac{\pi}{T_i} = \Omega_{ci}$, for signal bandwidth limits Ω_{ci} , will work fine.

Proof We start by writing $x(n_1, n_2, n_3)$ in terms of the samples of the inverse Fourier transform of $X_c(\Omega_1, \Omega_2, \Omega_3)$:

$$\begin{aligned} x(n_1, n_2, n_3) &= \frac{1}{(2\pi)^3} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} X_c(\Omega_1, \Omega_2, \Omega_3) \\ &\quad \times \exp + j(\Omega_1 n_1 T_1 + \Omega_2 n_2 T_2 + \Omega_3 n_3 T_3) d\Omega_1 d\Omega_2 d\Omega_3. \end{aligned}$$

Next, we let $\omega_i \triangleq \Omega_i T_i$ in this triple integral to obtain

$$\begin{aligned} x(n_1, n_2, n_3) &= \frac{1}{(2\pi)^3} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{1}{T_1 T_2 T_3} X_c \left(\frac{\omega_1}{T_1}, \frac{\omega_2}{T_2}, \frac{\omega_3}{T_3} \right) \\ &\quad \times \exp + j(\omega_1 n_1 + \omega_2 n_2 + \omega_3 n_3) d\omega_1 d\omega_2 d\omega_3 \\ &= \frac{1}{(2\pi)^3} \sum_{\text{all } k_1, k_2, k_3} \int_{SQ(k_1, k_2, k_3)} \frac{1}{T_1 T_2 T_3} X_c \left(\frac{\omega_1}{T_1}, \frac{\omega_2}{T_2}, \frac{\omega_3}{T_3} \right) \\ &\quad \times \exp + j(\omega_1 n_1 + \omega_2 n_2 + \omega_3 n_3) d\omega_1 d\omega_2 d\omega_3, \end{aligned} \quad (10.2-1)$$

where $SQ(k_1, k_2, k_3)$ is a $2\pi \times 2\pi \times 2\pi$ cube centered at position $(2\pi k_1, 2\pi k_2, 2\pi k_3)$:

$$\begin{aligned} SQ(k_1, k_2, k_3) &\triangleq [-\pi + 2\pi k_1, +\pi + 2\pi k_1] \times [-\pi + 2\pi k_2, +\pi + 2\pi k_2] \\ &\quad \times [-\pi + 2\pi k_3, +\pi + 2\pi k_3]. \end{aligned}$$

Then making the change of variables $\omega'_i = \omega_i - 2\pi k_i$ inside each of the preceding integrals, we get

$$\begin{aligned}
 x(n_1, n_2, n_3) &= \frac{1}{(2\pi)^2} \sum_{\text{all } k_1, k_2, k_3} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \frac{1}{T_1 T_2 T_3} \\
 &\quad \times X_c \left(\frac{\omega'_1 + 2\pi k_1}{T_1}, \frac{\omega'_2 + 2\pi k_2}{T_2}, \frac{\omega'_3 + 2\pi k_3}{T_3} \right) \\
 &\quad \times \exp + j(\omega'_1 n_1 + \omega'_2 n_2 + \omega'_3 n_3) d\omega'_1 d\omega'_2 d\omega'_3 \\
 &= \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \left[\sum_{\text{all } k_1, k_2} \frac{1}{T_1 T_2 T_3} \right. \\
 &\quad \times X_c \left(\frac{\omega'_1 + 2\pi k_1}{T_1}, \frac{\omega'_2 + 2\pi k_2}{T_2}, \frac{\omega'_3 + 2\pi k_3}{T_3} \right) \left. \right] \\
 &\quad \times \exp + j(\omega'_1 n_1 + \omega'_2 n_2 + \omega'_3 n_3) d\omega'_1 d\omega'_2 d\omega'_3 \\
 &= \text{IFT} \left[\sum_{\text{all } k_1, k_2, k_3} \frac{1}{T_1 T_2 T_3} X_c \left(\frac{\omega'_1 + 2\pi k_1}{T_1}, \frac{\omega'_2 + 2\pi k_2}{T_2}, \frac{\omega'_3 + 2\pi k_3}{T_3} \right) \right],
 \end{aligned}$$

as was to be shown. ■

This last equation says that x is the 3-D discrete-space IFT of X . If any T_i is too large, then this linear sampling theory says we should prefilter before sampling to avoid aliasing caused by the offensively coarse T_i 's.

General 3-D Sampling

Consider more general, but still regular sampling at locations

$$\begin{aligned}
 \mathbf{t} &= \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \mathbf{V} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \\
 &= n_1 \mathbf{v}_1 + n_2 \mathbf{v}_2 + n_3 \mathbf{v}_3,
 \end{aligned}$$

where $\mathbf{V} \triangleq [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ is the 3-D *sampling matrix* formed from three noncolinear sampling basis vectors $\mathbf{v}_1, \mathbf{v}_2$, and \mathbf{v}_3 . The corresponding aliased representation in the discrete-space frequency domain is expressed in terms of the *periodicity matrix* \mathbf{U} , which must satisfy

$$\mathbf{U}^T \mathbf{V} = 2\pi \mathbf{I}.$$

Just as in the 2-D case, one can show that

$$X(\boldsymbol{\omega}) = \frac{1}{|\det \mathbf{V}|} \sum_{\mathbf{k}} X_c \left[\frac{1}{2\pi} \mathbf{U}(\boldsymbol{\omega} - 2\pi \mathbf{k}) \right], \quad (10.2-2)$$

where X_c is the 3-D continuous-space Fourier transform, and X is the 3-D Fourier transform of the samples. We can also express (10.2-2) in terms of the analog frequency $\mathbf{\Omega}$ as

$$X(\mathbf{V}^T \mathbf{\Omega}) = \frac{1}{|\det \mathbf{V}|} \sum_k X_c(\mathbf{\Omega} - \mathbf{U}k),$$

analogously to the 2-D general sampling case in Chapter 2.

When X_c is bandlimited, and in the absence of aliasing, we have

$$X_c(\mathbf{\Omega}) = |\det \mathbf{V}| X(\mathbf{V}^T \mathbf{\Omega}),$$

which we can use for reconstruction from the 3-D samples. We start with the 3-D continuous-space IFT relation

$$x_c(\mathbf{t}) = \frac{1}{(2\pi)^3} \int \int \int X_c(\mathbf{\Omega}) \exp(j\mathbf{\Omega}^T \mathbf{t}) d\mathbf{\Omega},$$

and substitute to obtain

$$\begin{aligned} x_c(\mathbf{t}) &= \frac{|\det \mathbf{V}|}{(2\pi)^3} \int \int \int X(\mathbf{V}^T \mathbf{\Omega}) \exp(j\mathbf{\Omega}^T \mathbf{t}) d\mathbf{\Omega} \\ &= \frac{|\det \mathbf{V}|}{(2\pi)^3} \int \int \int \sum_n x(\mathbf{n}) \exp(-j\mathbf{\Omega}^T \mathbf{V}\mathbf{n}) \exp(j\mathbf{\Omega}^T \mathbf{t}) d\mathbf{\Omega} \\ &= \sum_n x(\mathbf{n}) \frac{|\det \mathbf{V}|}{(2\pi)^3} \int \int \int \exp(j\mathbf{\Omega}^T (\mathbf{t} - \mathbf{V}\mathbf{n})) d\mathbf{\Omega}. \end{aligned}$$

We thus see that the reconstruction interpolation filter is

$$h(\mathbf{t}) = \frac{|\det \mathbf{V}|}{(2\pi)^3} \int \int \int \exp(j\mathbf{\Omega}^T \mathbf{t}) d\mathbf{\Omega},$$

where the 3-D integral is taken over the bandlimited support of X_c , say $\|\mathbf{\Omega}\| \leq \Omega_c$. We then have the reconstruction formula

$$x_c(\mathbf{t}) = \sum_n x(\mathbf{n}) h(\mathbf{t} - \mathbf{V}\mathbf{n}).$$

For more on general 3-D sampling and sample rate change, see the text by Vaidyanathan [1].

10.3 SPATIOTEMPORAL SIGNAL PROCESSING

In spatiotemporal processing usually data are undersampled in the time dimension *and* there is no prefilter! The situation should improve as technology permits higher frame rates than 24–30 frames per second (fps), such as 60–100 fps, etc. Here, we specialize 3-D processing to two spatial dimensions and one time dimension. We can then treat the filtering of image sequences or video. Notationally we replace n_3 by n ,

which will denote discrete time. Also, the third frequency variable is written simply as ω radians or f when the unit is Hz, giving the spatiotemporal FT pair

$$x(n_1, n_2, n) \leftrightarrow X(\omega_1, \omega_2, \omega).$$

Time-domain causality now applies in the third dimension n . We can use a matrix notation

$$\mathbf{x}(n) \triangleq \{x(\mathbf{n})\} \quad \text{with} \quad \mathbf{n} \triangleq (n_1, n_2, n)^T$$

to denote a frame of video data at time n . The video is then a sequence of these matrices:

$$\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \mathbf{x}(3), \mathbf{x}(4), \mathbf{x}(5), \dots$$

Spatiotemporal Sampling

Example 10.3–1: Progressive Video

We apply rectangular sampling to the spatiotemporal continuous signal $x_c(t_1, t_2, t)$, sampled spatially with sample period $T_1 \times T_2$ and temporally with period Δt . We then have the sampling and periodicity matrices,

$$\mathbf{V} = \begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} 2\pi/T_1 & 0 & 0 \\ 0 & 2\pi/T_2 & 0 \\ 0 & 0 & \frac{2\pi}{\Delta t} \end{bmatrix}.$$

This spatiotemporal sampling is called *progressive* scanning, or simply *noninterlaced*, by video engineers.

Computer manufacturers have long favored progressive, also called noninterlaced, because small characters do not flicker as much on a computer display. For natural video, progressive avoids the interline flicker of interlaced systems and is easier to interface with motion estimation and compensation schemes, which save channel bits. Nevertheless, interlaced CRT displays are cheaper to build, because the pixel rate is lower for interlace given the same display resolution. Now solid-state displays are most all progressive.

Example 10.3–2: Interlaced Video

In Chapter 2, we introduced this concept for analog video as just 2-D sampling in the vertical and time directions $(v, t) = (t_2, t)$, with sampling and periodicity matrices,

$$\mathbf{V} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} \pi & \pi \\ \pi & -\pi \end{bmatrix}.$$

It is only 2-D sampling because in analog video there is no sampling horizontally. However, in digital video, there is horizontal sampling, so we need 3-D sampling. The interlaced

digital video has sampling matrix in space-time and periodicity matrix in spatiotemporal frequency given as

$$\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{U} = \begin{bmatrix} 2\pi & 0 & 0 \\ 0 & \pi & \pi \\ 0 & \pi & -\pi \end{bmatrix},$$

in terms of sampling structure $(h, v, t) = (t_1, t_2, t)$. Here, we have normalized the sampling periods for simplicity, and the resulting simple sampling pattern is called *quincunx*. Of course, in a real system, the sample spacings would not be unity, and we would have sampling in terms of distances T_1 and T_2 on the frame, and T as the time between fields.²

Spatiotemporal frequency aliasing is a real problem in video signal processing because optical lenses are poor bandlimiting filters and, further, the temporal dimension is usually grossly undersampled (i.e., 30 fps is often not high enough). A frame rate of 100 fps or more has been often claimed as necessary for common scenes. Only now with HD cameras and beyond is digital video beginning to become alias free in space. Alias-free with respect to time sampling, however, remains elusive, without the use of special high frame-rate cameras. They are used in certain industrial inspection applications and in the movie industry to create slow motion effects.

Spatiotemporal Filters

The variety of spatiotemporal filters includes finite impulse response (FIR), infinite impulse response (IIR), statistically designed filters, and adaptive filters. Additionally, there can be hybrids of FIR and IIR (e.g., the filter can be IIR in time and FIR in space). This kind of hybrid filter has become very popular due to the need to minimize memory in the temporal direction, thus minimizing the use of *frame memories*. The frame predictor in a hybrid video coder (cf. Chapter 12) is an FIR spatiotemporal filter. The decoder must run the inverse of this filter, which is thus an IIR spatiotemporal filter. It must therefore be 3-D multidimensionally stable.

Example 10.3–3: Temporal Averager

A very simple example of a useful spatiotemporal filter is the temporal averager, whose impulse response is given as

$$h(n_1, n_2, n) = \frac{1}{N} \delta(n_1, n_2) (u(n) - u(n - N)).$$

²We remember from Chapter 2 that an interlaced frame is composed of two fields, the upper and lower.

For input $x(n_1, n_2, n)$, the output $y(n_1, n_2, n)$ is given as

$$y(n_1, n_2, n) = \frac{1}{N} \sum_{k=0}^{N-1} x(n_1, n_2, n-k),$$

or just the average of the present and past $N - 1$ frames. This is a 3-D FIR filter that is causal in the time or n variable. Such a filter is often used to smooth noisy video data, most often due to image sensor noise, when the frames are very similar i.e., have little interesting change over the time scale of N frames. ■

If objects in the frame move much, then the temporal averager will blur their edges, so often such a device is combined with a suitable motion or change detector that will control the number of frames actually used. It can be based on some norm of the frame difference $\|x(n_1, n_2, n) - x(n_1, n_2, n-1)\|$. Optimal filtering of noisy image sequences can be based on statistical models of the noise and data. One such model that is quite common is the spatiotemporal autoregressive moving average (ARMA) *random field sequence* (i.e., a sequence of random fields), which we discuss next.

Spatiotemporal ARMA Model

Definition 10.3–1: Autoregressive Moving Average

$$\begin{aligned} x(n_1, n_2, n) = & \sum_{(k_1, k_2, k) \in \mathcal{R}_a - (0,0,0)} a_{k_1, k_2, k} x(n_1 - k_1, n_2 - k_2, n - k) \\ & + \sum_{(k_1, k_2, k) \in \mathcal{R}_b} b_{k_1, k_2, k} w(n_1 - k_1, n_2 - k_2, n - k) \end{aligned} \quad (10.3-1)$$

Here, the input spatiotemporal sequence w is taken as a white noise random-field sequence. If $\mathcal{R}_b = \{(0,0,0)\}$, then the model is AR, or autoregressive. We get a Markov random field sequence (cf. [Section 10.4](#)) if the input w is jointly independent as well as spectrally white. If additionally \mathcal{R}_a only has coefficients for $t \geq 1$, then the model is said to be *frame-based*. A frame-based model could use parallel computation capability to calculate a whole frame at a time.

Intraframe Filtering

The equation for intraframe filtering with filter impulse response $h(n_1, n_2)$ is

$$y(n_1, n_2, n) = \sum_{k_1, k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2, n).$$

Intraframe filtering is appropriate to restore mild spatial blur in the presence of good SNR, i.e., 30 dB or higher, and also when there is low interframe dependency, (e.g., a very low frame rate such as < 1fps). Still other reasons are to retain frame identity, such as in video editing, using so-called *I* frames in MPEG parlance,³ to reduce complexity and expense.

The *intraframe ARMA signal model* is given as

$$\begin{aligned} x(n_1, n_2, n) = & \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2} x(n_1 - k_1, n_2 - k_2, n) \\ & + \sum_{(k_1, k_2) \in \mathcal{R}_b} b_{k_1, k_2} w(n_1 - k_1, n_2 - k_2, n). \end{aligned} \quad (10.3-2)$$

Normally the input term w is taken as spatiotemporal white noise. This ARMA consists of spatial filtering of each input frame, with the 2-D recursive filter with coefficient arrays $\{a\}, \{b\}$. The filter coefficients are constant here across the frames but could easily be time variant $\{a(k_1, k_2, k)\}, \{b(k_1, k_2, k)\}$, making a time-varying intraframe ARMA model.

Intraframe Wiener Filter

Taking the 2-D Fourier transform of (10.3-2), we obtain

$$X(\omega_1, \omega_2; n) = \frac{B(\omega_1, \omega_2)}{1 - A(\omega_1, \omega_2)} W(\omega_1, \omega_2; n),$$

for each frame (n). We could also write this in terms of 3-D Fourier transforms as

$$X(\omega_1, \omega_2, \omega) = \frac{B(\omega_1, \omega_2)}{1 - A(\omega_1, \omega_2)} W(\omega_1, \omega_2, \omega)$$

but will generally reserve 3-D transform notation for the interframe case to be treated in the following subsections.

For homogeneous random processes that possess power spectral densities (PSDs), we can write

$$\begin{aligned} S_x(\omega_1, \omega_2; n) &= \left| \frac{B(\omega_1, \omega_2)}{1 - A(\omega_1, \omega_2)} \right|^2 S_w(\omega_1, \omega_2; n) \quad \text{and} \\ S_x(\omega_1, \omega_2, \omega) &= \left| \frac{B(\omega_1, \omega_2)}{1 - A(\omega_1, \omega_2)} \right|^2 S_w(\omega_1, \omega_2, \omega), \end{aligned}$$

where often $S_w = \sigma_w^2$ (i.e., spatiotemporal white noise). Here, A and B are the spatial or 2-D Fourier transforms of the spatial filter coefficients $a(k_1, k_2)$ and $b(k_1, k_2)$.

³*I-frame* is an MPEG compression term that denotes an image frame compressed without reference to other image frames (i.e., intraframe compression). MPEG compression is covered in Chapter 12.

We can then use these spectral densities to design a 2-D or spatial Wiener filter for the observations

$$y(n_1, n_2, n) = h(n_1, n_2) * x(n_1, n_2, n) + v(n_1, n_2, n),$$

where the spatial blur h is known and the noise v is white with variance σ_v^2 and uncorrelated with the signal x . Then the *intraframe Wiener filter* is given as

$$G(\omega_1, \omega_2; n) = \frac{H^*(\omega_1, \omega_2) S_x(\omega_1, \omega_2; n)}{|H|^2(\omega_1, \omega_2) S_x(\omega_1, \omega_2; n) + \sigma_w^2} \quad (10.3-3)$$

at each frame n , where the PSDs S_x could have been estimated via an AR or ARMA modeling procedure. Effectively, this is just image processing done for each frame (n). Note that the PSDs can vary with n , i.e., be time variant.

Example 10.3–4: Intraframe Wiener Filter

Let $r = x + v$ with $x \perp v$ (i.e., x and v orthogonal). Also assume that the random field sequences x and n are homogeneous in space with PSDs S_x and S_v , respectively. Then the noncausal intraframe Wiener filter for frame n is given by

$$G(\omega_1, \omega_2; n) = \frac{S_x(\omega_1, \omega_2; n)}{S_x(\omega_1, \omega_2; n) + S_v(\omega_1, \omega_2; n)}.$$

Additionally, assume a spatial AR model for the signal x as in (10.3–1) and also that the observation noise is white with PSD σ_v^2 . Then the intraframe Wiener filter becomes

$$\begin{aligned} G(\omega_1, \omega_2; n) &= \frac{\left| \frac{B(\omega_1, \omega_2)}{1-A(\omega_1, \omega_2)} \right|^2 \sigma_w^2}{\left| \frac{B(\omega_1, \omega_2)}{1-A(\omega_1, \omega_2)} \right|^2 \sigma_w^2 + \sigma_v^2} \\ &= \frac{|B(\omega_1, \omega_2)|^2 \sigma_w^2}{|B(\omega_1, \omega_2)|^2 \sigma_w^2 + |1-A(\omega_1, \omega_2)|^2 \sigma_v^2}. \end{aligned}$$

In the case of Wiener filtering, the optimality of this process would, of course, require that the frames be uncorrelated, both for the signal x and for the noise w . This is even though the blur h is confined to the individual frames—i.e., $h = h(n_1, n_2)$ —which is also, of course, a necessary condition for the optimality of the linear filter (10.3–3).

Interframe Filtering

The equation for interframe filtering with filter impulse response $h(n_1, n_2, n)$ is

$$y(n_1, n_2, n) = \sum_{k_1, k_2} h(k_1, k_2, n) x(n_1 - k_1, n_2 - k_2, n - k).$$

Interframe filtering is the general case for spatiotemporal LSI processing and is the basis for powerful motion-compensated models to be treated in Chapter 11.

The *interframe ARMA signal model* is given as

$$\begin{aligned} x(n_1, n_2, n) = & \sum_{(k_1, k_2, k) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2, k} x(n_1 - k_1, n_2 - k_2, n - k) \\ & + \sum_{(k_1, k_2, k) \in \mathcal{R}_b} b_{k_1, k_2, k} w(n_1 - k_1, n_2 - k_2, n - k). \end{aligned} \quad (10.3-4)$$

Often these interframe models are categorized by how many prior frames they use. If we restrict to just one prior frame, we can write

$$\begin{aligned} x(n_1, n_2, n) = & \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2, 0} x(n_1 - k_1, n_2 - k_2, n) \\ & + \sum_{(k_1, k_2) \in \mathcal{R}_a - (0,0)} a_{k_1, k_2, 1} x(n_1 - k_1, n_2 - k_2, n - 1) \\ & + \sum_{(k_1, k_2) \in \mathcal{R}_b} b_{k_1, k_2, 0} w(n_1 - k_1, n_2 - k_2, n) \\ & + \sum_{(k_1, k_2) \in \mathcal{R}_b} b_{k_1, k_2, 1} w(n_1 - k_1, n_2 - k_2, n - 1), \end{aligned} \quad (10.3-5)$$

which explicitly shows the operation as FIR spatial filterings of the present and past input and output frames, and then their output summation to produce the next output point in the current frame.

In the frequency domain, we can write this filtering using 3-D Fourier transforms as

$$X(\omega_1, \omega_2, \omega) = \frac{B(\omega_1, \omega_2, \omega)}{1 - A(\omega_1, \omega_2, \omega)} W(\omega_1, \omega_2, \omega),$$

in the general case of (10.3-4) and

$$X(\omega_1, \omega_2, \omega) = \frac{B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}}{1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}} W(\omega_1, \omega_2, \omega),$$

in the first-order temporal case of (10.3-5).

Interframe Wiener Filter

For the more general observation equation, with both blurring and observation noise,

$$y(n_1, n_2, n) = h(n_1, n_2, n) * x(n_1, n_2, n) + v(n_1, n_2, n),$$

we will need the *interframe Wiener filter*, expressed in 3-D Fourier transforms as

$$G(\omega_1, \omega_2, \omega) = \frac{H^*(\omega_1, \omega_2, \omega) S_x(\omega_1, \omega_2, \omega)}{|H|^2(\omega_1, \omega_2, \omega) S_x(\omega_1, \omega_2, \omega) + \sigma_v^2},$$

where we see the 3-D PSD $S_x(\omega_1, \omega_2, \omega)$ and the observation noise v is here assumed to be uncorrelated in time as well as in space, i.e., $R_v(m_1, m_2, m) = \sigma_v^2 \delta(m_1, m_2, m)$, with δ the 3-D Kronecker delta function.

Example 10.3–5: First-Order Temporal Interframe Wiener Filter

For the case of the first-order temporal ARMA model of (10.3–5), we can compute the PSD as

$$S_x(\omega_1, \omega_2, \omega) = \left| \frac{B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}}{1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}} \right|^2 \sigma_w^2,$$

so that the resulting interframe Wiener filter becomes

$$\begin{aligned} G(\omega_1, \omega_2, \omega) &= \frac{H^*(\omega_1, \omega_2, \omega) \left| \frac{B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}}{1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}} \right|^2 \sigma_w^2}{|H|^2(\omega_1, \omega_2, \omega) \left| \frac{B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}}{1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}} \right|^2 \sigma_w^2 + \sigma_v^2} \\ &= \frac{H^*(\omega_1, \omega_2, \omega) |B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}|^2 \sigma_w^2}{|H|^2(\omega_1, \omega_2, \omega) |B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}|^2 \sigma_w^2 + |1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}|^2 \sigma_v^2}. \end{aligned}$$

We notice that when $|H|^2(\omega_1, \omega_2, \omega) |B_0(\omega_1, \omega_2) + B_1(\omega_1, \omega_2)e^{-j\omega}|^2 \sigma_w^2 \gg |1 - A_0(\omega_1, \omega_2) - A_1(\omega_1, \omega_2)e^{-j\omega}|^2 \sigma_v^2$, then G is very close to an inverse filter. This condition is just the case where the blurred signal PSD $|H|^2 S_x \gg \sigma_v^2$, the power density of the white observation noise. ■

More on intraframe and interframe Wiener filters, along with application to both estimation and restoration, is contained in the text by Tekalp [2].

In a 2-D ARMA model, if the input coefficient filtering through $b_{k_1, k_2, k}$ is absent, we have just the 2-D NSHP AR model, which can be NSHP Markov in the case where the input random field is i.i.d. In particular, if the input sequence w is white Gaussian, then the resulting random field is NSHP Markov. Next we turn to spatiotemporal or 3-D Markov.

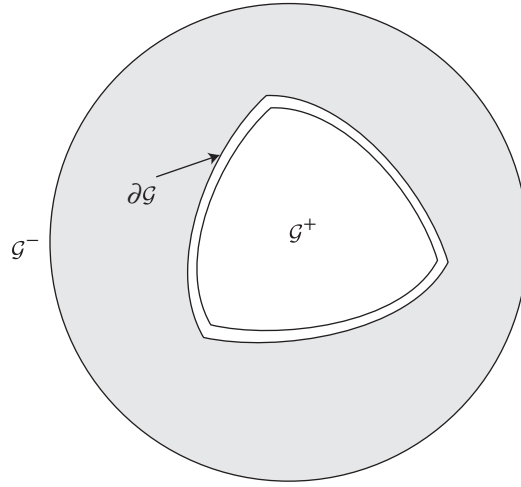
10.4 SPATIOTEMPORAL MARKOV MODELS

We start out with the general definition of Markov random field sequence.

Definition 10.4–1: Discrete Markov Random Field Sequence

Let x be a random field sequence on \mathcal{Z}^3 , the 3-D lattice. Let a band of minimum width p , $\partial\mathcal{G}$ (“the present”) separate \mathcal{Z}^3 into two regions: \mathcal{G}^+ (“the future”) and \mathcal{G}^- (“the past”). Then x is Markov- p if $\{x|_{\mathcal{G}^+} \text{ given } x|_{\partial\mathcal{G}}\}$ is independent of $\{x|_{\mathcal{G}^-}\}$ for all $\partial\mathcal{G}$. ■

These regions are illustrated in Figure 10.4–1, which shows a spherical region \mathcal{G}^+ inside a sphere of thickness $\partial\mathcal{G}$. Outside the sphere is the region \mathcal{G}^- .


FIGURE 10.4-1

Noncausal Markov field regions. \mathcal{G}^+ is the region inside a sphere of thickness $\partial\mathcal{G}$.

For the homogeneous, zero-mean Gaussian case, this noncausal Markov definition can be expressed by the following recursive equation model:

$$x(n_1, n_2, n) = \sum_{(k_1, k_2, k) \in D_p} c(k_1, k_2, k) x(n_1 - k_1, n_2 - k_2, n - k) + u(n_1, n_2, n), \quad (10.4-1)$$

where

$$E\{x(n_1, n_2, n) u(k_1, k_2, k)\} = \sigma_u^2 \delta_{n_1, k_1} \delta_{n_2, k_2} \delta_{n, k},$$

$$D_p \triangleq \{n_1, n_2, n | n_1^2 + n_2^2 + n^2 \leq p^2 \text{ and } (n_1, n_2, n) \neq (0, 0, 0)\}.$$

Here, $u(n_1, n_2, n)$ is a Gaussian, zero-mean, homogeneous random-field sequence with correlation function of bounded support,

$$R_u(n_1, n_2, n) = \begin{cases} \sigma_u^2, & (n_1, n_2, n) = \mathbf{0}, \\ -c_{k_1, k_2, n} \sigma_u^2, & (n_1, n_2, n) \in D_p - \mathbf{0}, \\ 0, & \text{elsewhere.} \end{cases}$$

The $c_{k_1, k_2, k}$ are the interpolation coefficients of the minimum mean-square error (MMSE) linear interpolation problem:

Given data x for $(n_1, n_2, n) \neq \mathbf{0}$, find the best linear estimate of $s(\mathbf{0})$. The solution is given as the conditional mean,

$$\begin{aligned} E\{x(\mathbf{0}) | x \text{ on } (n_1, n_2, n) \neq \mathbf{0}\} \\ = \sum_{(k_1, k_2, k) \in D_p} c(k_1, k_2, k) x(-k_1, -k_2, -k). \end{aligned}$$

In this formulation, σ_u^2 is the mean-square interpolation error.

Actually, in Fig. 10.4–1 we could swap the past and future, \mathcal{G}^- and \mathcal{G}^+ , and this would correspond to running a 1-D Markov process either forward or backward in time. However, as labeled in the Figure, with the ‘future’ inside, when we shrink the sphere down to just one interior point, we get the interpolative model (10.4–1).

Causal and Semicausal 3-D Field Sequences

The future \mathcal{G}^+ can be defined in various ways that are constrained by the support of the coefficient array $\text{supp}(c)$:

- $\{n > 0\}$ in the *temporally causal case*.
- $\{(n_1, n_2) \neq (0, 0), n = 0\} \cup \{n > 0\}$ in the *temporally semicausal case*.
- $\{n_1 \geq 0, n_2 \geq 0, n = 0\} \cup \{n_1 < 0, n_2 > 0, n = 0\} \cup \{n > 0\}$, the *totally ordered temporally causal case*, also called a *nonsymmetric half-space (NSHS)*.

The future-present-past diagram of the temporally causal model is shown in Figure 10.4–2. It can be obtained from the noncausal model by stretching out the radius of the sphere in Figure 10.4–1 to infinity. Then the thickness of the sphere becomes the plane shown in the temporally causal Figure 10.4–2 for the Markov-1 case. A first-order temporally causal model can be constructed as follows:

$$x(n_1, n_2, n) = \sum_{k_1, k_2} c(k_1, k_2, 1)x(n_1 - k_1, n_2 - k_2, n - 1) + u(n_1, n_2, n),$$

$$u(n_1, n_2, n) = \sum_{k_1, k_2} c^1(k_1, k_2, 1)x(n_1 - k_1, n_2 - k_2, n) + u^1(n_1, n_2, n).$$

Here, $u(n_1, n_2, n)$ is uncorrelated in the time direction only, and $u^1(n_1, n_2, n)$ is strict near-neighbor correlated in space but completely uncorrelated in time. Effectively, we have a sequence of 2-D Gauss-Markov⁴ random fields u , driving a frame-wise predictive recursion for x .

The future-present-past diagram for a temporally semicausal model is shown in Figure 10.4–3.

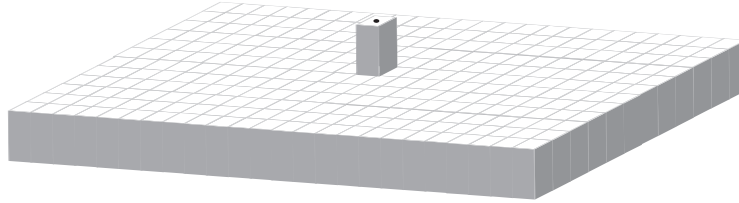
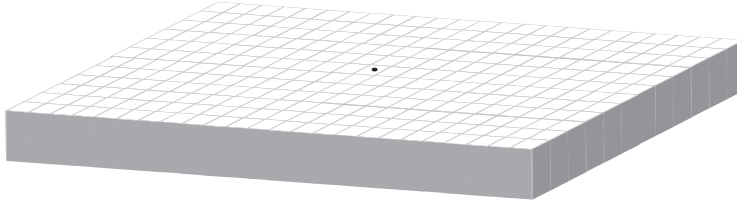


FIGURE 10.4–2

Temporally causal. The dot (·) indicates the present, and the plane below is the immediate past.

⁴The field x becomes Markov in the Gaussian case. Otherwise, it is only wide-sense Markov.


FIGURE 10.4-3

Temporally semicausal. The dot (\cdot) indicates the present, and it is surrounded by the immediate past.

We can construct a Markov p th-order temporal example:

$$x(n_1, n_2, n) = \sum_{D_p} c(k_1, k_2, k) x(n_1 - k_1, n_2 - k_2, n - k) + u(n_1, n_2, n),$$

where

$$R_u(n_1, n_2, n) = \begin{cases} \sigma_u^2, & n = 0 \text{ and } (n_1, n_2) = \mathbf{0}, \\ -c_{k_1, k_2, t} \sigma_u^2, & n = 0 \text{ and } (n_1, n_2) \in D_p - \mathbf{0}, \\ 0, & n \neq 0 \text{ or } (n_1, n_2) \notin D_p. \end{cases}$$

The reader should note that the classification of 3-D field sequences introduced here goes beyond Markov models. It is generally useful to consider spatiotemporal models as belonging to one of the three classes: temporally causal, temporally semicausal, and totally ordered temporally causal, as introduced in this section.

Reduced Update Spatiotemporal Kalman Filter

For a 3-D recursive filter, we must separate the past from future of the 3-D random field sequence. One way is to assume that the random field sequence is scanned in line-by-line and frame-by-frame (i.e., the totally ordered temporally causal case):

$$x(n_1, n_2, n) = \sum_{(k_1, k_2, k) \in S_c} c(k_1, k_2, k) x(n_1 - k_1, n_2 - k_2, n - k) + w(n_1, n_2, n),$$

where $w(n_1, n_2, n)$ is a white noise field sequence, with variance σ_w^2 , and where $S_c = \{n_1 \geq 0, n_2 \geq 0, n = 0\} \cup \{n_1 < 0, n_2 > 0, n = 0\} \cup \{n > 0\}$.

Our observed image model is given by

$$r(n_1, n_2, n) = \sum_{(k_1, k_2, k) \in S_h} h(k_1, k_2, k) x(n_1 - k_1, n_2 - k_2, n - k) + v(n_1, n_2, n).$$

The region S_h is the support of the spatiotemporal psf $h(k_1, k_2, k)$, which can model a blur or other distortion extending over multiple frames. The observation noise $v(n_1, n_2, n)$ is assumed to be an additive, zero-mean, homogeneous Gaussian field with covariance $Q_v(n_1, n_2, n) = \sigma_v^2 \delta(n_1, n_2, n)$, i.e., a white Gaussian noise.

In recursive filtering, only a finite subset of the NSHS is updated at each step, called the *update region*. This update region is slightly enlarged from the model support or local state for an (M_1, M_2, M) th-order $\oplus \oplus +$ model of the NSHS variety.

Local State and Update Region

The local state vector of 3-D spatiotemporal RUKF that is of order $M = 1$ in the temporal direction, and spatial order $M_1 = M_2$, is given as

$$\begin{aligned} \mathbf{x}(n_1, n_2, n) = & [x(n_1, n_2, n), \dots, x(n_1 - M_1 + 1, n_2, n); \\ & x(n_1 + M_1 + 1, n_2 - 1, n), \dots, s(n_1 - M_1 + 1, n_2 - 1, n); \\ & \dots; x(n_1 + M_1 + 1, n_2 - M_2, n), \dots, x(n_1 - M_1 + 1, n_2 - M_2, n); \\ & x(n_1 + M_1 + 1, n_2 + M_2, n - 1), \dots, x(n_1 - M_1 + 1, n_2 + M_2, n - 1); \\ & \dots; x(n_1 + M_1 + 1, n_2 - M_2, n - 1), \dots, x(n_1 - M_1 + 1, n_2 - M_2, n - 1)]^T. \end{aligned}$$

Figure 10.4–4 shows a $1 \times 1 \times 1$ -order local state region as the dark pixels, with a $2 \times 2 \times 1$ -order update region $\mathcal{U}_{\oplus\oplus+}(n_1, n_2, n)$, to be updated recursively, shown as the medium dark pixels in the figure. The total region shown is the covariance update support region $\mathcal{T}_{\oplus\oplus+}(n_1, n_2, n)$. We see that these regions are NSHP in the current frame (the top layer) but of a symmetric support in the previous frame (lower layer). We only show two frames here, but, of course, there could be more, which would constitute a higher order recursive processing. The current pixel location in the current frame (n_1, n_2, n) is shown as the black dot in the figure.

The resulting approximate RUKF equations are given as

$$\begin{aligned} \hat{x}_b(n_1, n_2, n) = & \sum_{(k_1, k_2, k) \in \mathcal{S}_c} c(k_1, k_2, k) \hat{x}_a(n_1 - k_1, n_2 - k_2, n - k), \\ \hat{x}_a(k_1, k_2, k) = & \hat{x}_b(k_1, k_2, k) + g^{(n_1, n_2, n)}(n_1 - k_1, n_2 - k_2, n - k) \\ & \times \left[r(n_1, n_2, n) - \sum_{(l_1, l_2, l) \in \mathcal{S}_h} h(l_1, l_2, l) \hat{x}_b(n_1 - l_1, n_2 - l_2, n - l) \right], \end{aligned}$$

for all $(k_1, k_2, k) \in \mathcal{U}_{\oplus\oplus+}(n_1, n_2, n)$, the current update region.

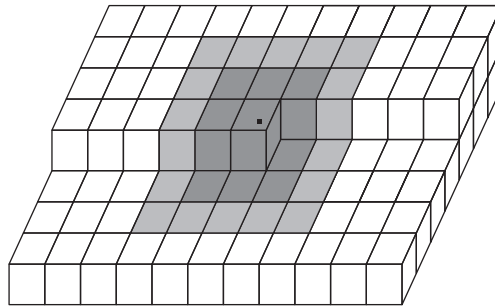


FIGURE 10.4–4

An illustration of 3-D Kalman-reduced support regions.

The gain factors $g^{(n_1, n_2, n)}(n_1 - k_1, n_2 - k_2, n - k)$ are given in terms of error covariance functions $R_b^{(n_1, n_2, n)}$ by

$$g^{(n_1, n_2, n)}(k_1, k_2, k) = \frac{\sum_{(l_1, l_2, l) \in S_h} h(l_1, l_2, l) R_b^{(n_1, n_2, n)}(n_1 - l_1, n_2 - l_2, n - l; n_1 - k_1, n_2 - k_2, n - k)}{\sum_{(l_1, l_2, l) \in S_h} \sum_{(o_1, o_2, o) \in S_h} h(l_1, l_2, l) R_b^{(n_1, n_2, n)}(n_1 - l_1, n_2 - l_2, n - l; n_1 - o_1, n_2 - o_2, n - o) + \sigma_v^2},$$

for all $(k_1, k_2, k) \in \mathcal{U}_{\oplus\oplus+}(n_1, n_2, n)$. The error-covariance equations are given recursively as, *before update*,

$$R_b^{(n_1, n_2, n)}(n_1, n_2, n; k_1, k_2, k) = \sum_{(l_1, l_2, l) \in S_c} c(l_1, l_2, l) R_a^{(n_1, n_2, n)}(n_1 - l_1, n_2 - l_2, n - l; k_1, k_2, k),$$

for all $(k_1, k_2, k) \in \mathcal{T}_{\oplus\oplus+}(n_1, n_2, n)$,

and

$$R_b^{(n_1, n_2, n)}(n_1, n_2, n; n_1, n_2, n) = \sum_{(l_1, l_2, l) \in S_c} c(l_1, l_2, l) R_a^{(n_1, n_2, n)}(n_1, n_2, n; n_1 - l_1, n_2 - l_2, n - l) + \sigma_w^2,$$

and *after update*,

$$R_a^{(n_1, n_2, n)}(k_1, k_2, k; l_1, l_2, l) = R_b^{(n_1, n_2, n)}(k_1, k_2, k; l_1, l_2, l) - g^{(n_1, n_2, n)}(n_1 - k_1, n_2 - k_2, n - k) \times \sum_{(o_1, o_2, o) \in S_h} h(o_1, o_2, o) R_b^{(n_1, n_2, n)}(n_1 - o_1, n_2 - o_2, n - o; l_1, l_2, l),$$

for all $(k_1, k_2, k) \in \mathcal{U}_{\oplus\oplus+}(n_1, n_2, n)$ and for all $(l_1, l_2, l) \in \mathcal{T}_{\oplus\oplus+}(n_1, n_2, n)$.

As processing proceeds into the data, and away from any spatial boundaries, stability of the model generally has been found to lead to convergence of these equations to a steady state fairly rapidly, under the condition that the covariance update region $\mathcal{T}_{\oplus\oplus+}(n_1, n_2, n)$ is sufficiently large. In that case, the superscripts on the error covariances and gain terms can be dropped, and we have simply

$$g(k_1, k_2, k) = \frac{\sum_{(l_1, l_2, l) \in S_h} h(l_1, l_2, l) R_b(n_1 - l_1, n_2 - l_2, n - l; n_1 - k_1, n_2 - k_2, n - k)}{\sum_{(l_1, l_2, l) \in S_h} \sum_{(o_1, o_2, o) \in S_h} h(l_1, l_2, l) h(o_1, o_2, o) R_b(n_1 - l_1, n_2 - l_2, n - l; n_1 - o_1, n_2 - o_2, n - o) + \sigma_v^2},$$

for all $(k_1, k_2, k) \in \mathcal{U}_{\oplus\oplus+}(n_1, n_2, n)$, in which case the 3-D RUKF reduces to a spatiotemporal LSI filter.

As in the 2-D and 1-D cases, we see that the nonlinear error-covariance equations do not involve the data, just the signal model and variance parameters. A good way to design the steady-state 3D RUKF is to run the error-covariance equations with the chosen stable signal model, not on the real image sequence, but just on a fictitious

but much smaller sequence. Typically, a sequence of 10 frames of 20×10 pixels is enough to obtain a quite accurate steady-state gain array g for commonly used image models. The 3-D RUKF has been extended to filter along motion trajectories [3, 4], and this version will be covered in the next chapter after introducing motion estimation.

CONCLUSIONS

This chapter has provided extensions to three dimensions of some of the earlier 2-D signal processing methods. In the important spatiotemporal case, we have provided results of a general nature for both deterministic and random models. Chapter 11 will apply these methods to the processing of the spatiotemporal signal that is video. Chapter 12 will provide applications of spatiotemporal processing in video compression and transmission.

PROBLEMS

1. Consider a $9 \times 9 \times 9$ -order FIR filter (i.e., 10 taps in each dimension). How many multiples are necessary per pixel to implement this filter in general? If 3-D separable? If each separable factor is linear phase?
2. In [Example 10.1–1](#), we wrote the separable convolution representation

$$y(n_1, n_2, n_3) = h_2(n_3) * [h_1(n_1, n_2) * x(n_1, n_2, n_3)]$$

of a system with two convolution operations. Carefully determine whether each one should be a 1-D or 2-D convolution by comparing to previous lines in this example.

3. Prove the 3-D Fourier transform property

$$\text{FT} \{x(\mathbf{n}) \exp + j\boldsymbol{\omega}_0^T \mathbf{n}\} = X(\boldsymbol{\omega} - \boldsymbol{\omega}_0),$$

and specialize to the case

$$\text{FT} \{(-1)^{n_1+n_2+n_3} x(\mathbf{n})\} = X(\omega_1 \pm \pi, \omega_2 \pm \pi, \omega_3 \pm \pi).$$

4. Extend the 2-D separable window filter design method of Chapter 5 to the 3-D case. Would there be any changes needed to the Kaiser window functions?
5. Extend [Example 10.3–2](#) to the case of interlaced 3-D sampling with general sampling distances T_1 and T_2 in space, and T in time. Display both the sampling matrix \mathbf{V} and the periodicity matrix \mathbf{U} .

6. If we sample the continuous-parameter function $x_c(t_1, t_2, t_3)$ with sample vectors

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{v}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

express the resulting discrete-parameter Fourier transform $X(\omega_1, \omega_2, \omega_3)$ in terms of the continuous-parameter Fourier transform $X_c(\Omega_1, \Omega_2, \Omega_3)$. Specialize your result to the case where there is no aliasing.

7. Consider the first noncausal Wiener filter design method (Section 8.2 of Chapter 8), and show that the method easily extends to the 3-D and spatiotemporal cases.
8. Estimate the memory size needed to run the 3-D RUKF equations on an image sequence of N frames, each frame being $N \times N$ pixels. Assume the model order is $1 \times 1 \times 1$, the update region is $2 \times 2 \times 1$, and the covariance update region is $4 \times 4 \times 1$.

REFERENCES

- [1] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [2] A. M. Tekalp, *Digital Video Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [3] J. W. Woods and J. Kim, "Motion Compensated Spatiotemporal Kalman Filter," Chapter 12 in *Motion Analysis and Image Sequence Processing*, Eds. I. Sezan and R. L. Lagendijk, Kluwer, Boston, MA, 1993.
- [4] J. Kim and J. W. Woods, "Spatiotemporal Adaptive 3-D Kalman Filter for Video," *IEEE Trans. Image Process.*, vol. 6, pp. 414–424, March 1997.

Digital Video Processing

11

In Chapter 10 we learned the generalization of multidimensional signal processing to the 3-D and spatiotemporal cases along with some relevant notation. In this chapter we apply and extend this material to processing the 3-D signal that is video—two spatial dimensions plus one time dimension. We look at the general *interframe* or cross-frame processing of digital video. We study the various methods of motion estimation for use in motion-compensated temporal filtering and motion-compensated extensions of 3-D Wiener and Kalman filters. We then look at the video processing applications of deinterlacing and frame-rate conversion. We next consider Bayesian methods for motion estimation and segmentation. We briefly consider the application of restoration of old film/video. We then look at joint motion estimation and segmentation. Finally, we consider super-resolution applied to video.

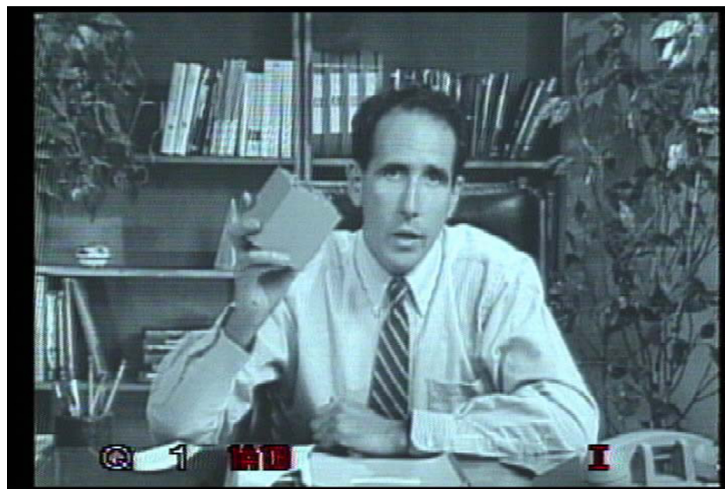
11.1 INTERFRAME PROCESSING

Interframe processing is a type of video processing of the present frame that makes use of input and processed video data from other frames. One area of application involves the estimation of a noisy video using a recursive filter that employs prior processed frames in its estimate of the current frame.

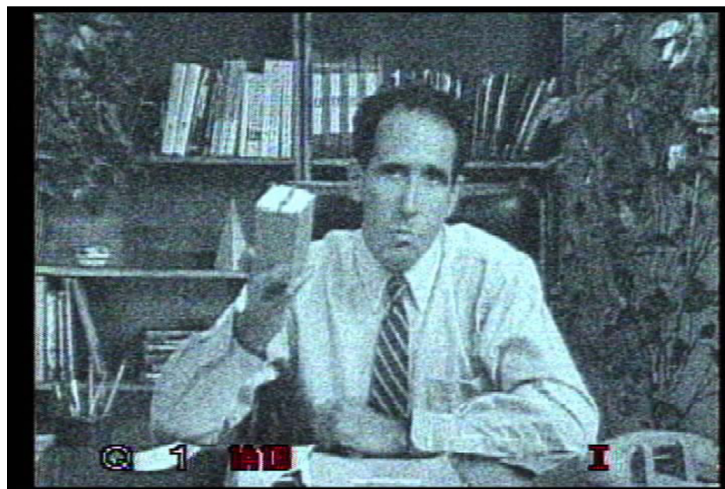
Example 11.1–1: Spatiotemporal RUKF

The spatiotemporal RUKF introduced in Chapter 10 has application in noisy image sequence processing. Again, the video scanning is assumed to be in the totally ordered temporal causal sense, and the RUKF processes the data in this scan-ordered sense. In an example from [1] the spatiotemporal RUKF processes a noisy version of the test clip *salesman* at the spatial resolution of 360×280 pixels, and the frame rate of 15 frames/sec (fps). We created the noisy input by adding Gaussian white noise to set up an input signal-to-noise ratio (SNR) of 10 dB. After processing, the SNR improved by 6 to 7 dB, to 16 to 17 dB after an initial start-up transient of approximately 10 frames.

We see a frame from the noise-free salesman clip in [Figure 11.1–1](#), followed by the noisy version in [Figure 11.1–2](#). The smoothed result from the spatiotemporal (3-D) RUKF

**FIGURE 11.1-1**

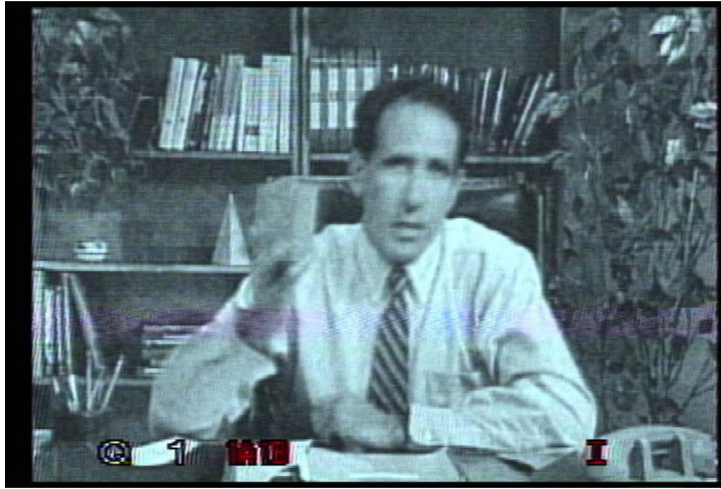
A frame from the original clip salesman.

**FIGURE 11.1-2**

A frame from the salesman clip with white noise added to achieve $\text{SNR} = 10 \text{ dB}$.

is shown in [Figure 11.1-3](#). We see that the white noise has been reduced and the image frame has become somewhat blurred or oversmoothed. ■

The limit to what can be achieved by LSI spatiotemporal filtering is determined by the overlap of the 3-D spectra of signal and noise. For fairly low-resolution signal and white noise, there will be a lot of overlap, and hence a smoothing of the signal is unavoidable. For higher resolution video, such as obtained from HD and SHD

**FIGURE 11.1-3**

A frame from the 3-D RUKF estimate.

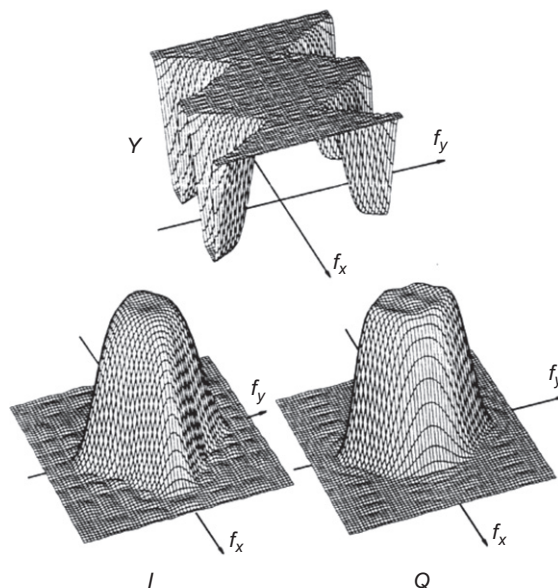
cameras or the high-resolution scanning of 35- and 70-mm film, the overlap of the signal and noise spectra can be much less. In that case, more impressive results can be obtained.

The next example shows how deterministic multidimensional filters can be used in processing analog standard definition television signals to obtain what is called *improved definition*. A little known fact is that the National Television System Committee (NTSC) color television lost some luma resolution that was present in the prior monochrome standard. Using multidimensional filters, it was possible to largely get back the lost resolution [2].

Example 11.1-2: Improved-Definition Television

The 3-D spectra of the old analog NTSC terrestrial broadcast television¹ is sketched in Figure 11.1-4, where the luma power is concentrated at low frequencies, and the chroma data have been modulated up to the sites shown via a so-called chroma subcarrier. Compositing the chroma data together with the luma data like this required, first lowpass filtering of the luma data, and then rather severe lowpass filtering of the chroma data. This was the compromise solution that the NTSC decided upon to provide compatible color television in mid-20th century America (that is compatible with the previously authorized monochrome, or luma-only TV). The downside of the NTSC *compatible color TV* was that the luma resolution went down slightly and the chroma (color) resolution was only about a sixth to an eighth of that of the luma (monochrome) resolution, and that led to rather

¹This is also known as a composite signal since bandlimited chroma data are composited together with the bandlimited luma data.

**FIGURE 11.1-4**

2-D filter responses for generalized NTSC encoding of Y , I , and Q components.
 ([2] © 1988 SMPTE)

blurry color. Further complicating matters, typical receivers of the time were not able to completely separate the two signals, leading to *cross-color* and *cross-luma* artifacts. In the mid-1990s, better solutions to separate the components at the transmitter were achieved through the use of multidimensional filtering. Perspective plots of three preprocessing filters are shown in Figure 11.1-4. The diamond band shape of these 11×31 -point finite impulse response (FIR) filters was obtained using a nonseparable design method by Dubois and Schreiber [2]. The Y, I, Q color space that was employed in NTSC was similar to the current Y', C_B, C_R color space used for digital TV coding and transmission, being a similar but somewhat different linear transformation on R', B', G' (see Section 6.5).

These filters could be employed prior to forming an NTSC composite signal with very little to no overlap of the luma and quadrature-modulated chroma components. The filters in Figure 11.1-4 marked I and Q would then be used to bandlimit the chroma components prior to their modulation up to fit in the spectral holes that the filter marked Y has created in the Y component. Together with appropriate 2-D postprocessing filters, cross-color and cross-luma artifacts were effectively eliminated [2]. ■

Receiver-processing multidimensional filters went by various names in the consumer electronics business, such as *comb* filter, *line comb*, and *frame comb*. Clearly, the best way to avoid these cross-contamination problems is to keep the luma and chroma signals separate, as in *RGB* and *YIQ component format*.

The next example shows modification of a designed spatial lowpass filter to meet spatial domain step-response characteristics.

Example 11.1–3: Video Processing Example (due to Schröder and Blume [3])

Video engineers are very concerned with the spatial impulse and step response of the filters used in video processing. In particular, they want the first overshoot of the step response to be close in to the main rising section and the further undershoots and overshoots to be small enough to be not visually noticed. One approach would be to use a general optimization program, extending the design methods of Chapter 5. First, choose an error criterion based on p th errors (p even) in both the frequency and space domains, with a weighting parameter λ to control the relative importance of frequency and spatial errors:

$$\|H_I(\omega_1, \omega_2) - H(\omega_1, \omega_2)\|_p + \lambda \|h_I(n_1, n_2) - h(n_1, n_2)\|_p.$$

However, a simpler procedure, due to Schröder [4], is to cascade an enhancement network with the lowpass filter that was conventionally designed. The goal of an enhancement circuit is to optimize the overall step response. The three-tap enhancement filter with impulse response

$$h_{\text{en}}(n_1) = -\frac{\alpha}{2}\delta(n_1 - 2) + (1 + \alpha)\delta(n_1) - \frac{\alpha}{2}\delta(n_1 + 2)$$

and frequency response

$$H_{\text{en}}(\omega_1) = 1 + \alpha - \alpha \cos 2\omega_1$$

can optimize the horizontal step response of a 2-D lowpass filter. The value of α is used to achieve the desired *visual definition* characteristics. Figure 11.1–5 shows a properly optimized step response on a vertical bar test pattern.



FIGURE 11.1–5

Properly optimized or compensated lowpass filtering of test signal. ([4] ©2000 John Wiley)

**FIGURE 11.1-6**

Nonoptimized filtering showing oversmoothing. ([4] © 2000 John Wiley)

**FIGURE 11.1-7**

Overoptimized filtering showing ringing. ([4] © 2000 John Wiley)

Figure 11.1-6 shows an underoptimized or nonoptimized result and displays an overly smoothed appearance. Figure 11.1-7 shows the result of an overly optimized filter (i.e., too sharp a transition band) showing ringing around the vertical edges. The same enhancement of the visual definition can be done in the vertical direction by adding the needed

vertical component to the enhancement filter, making the overall filter separable with 3×3 support. 

11.2 MOTION ESTIMATION AND MOTION COMPENSATION

Motion compensation (MC) is very useful in video filtering to remove noise and enhance signal. It is useful since it allows the filter or coder to process through the video on a path of near-maximum correlation based on following motion trajectories across the frames making up the image sequence or video. Motion compensation is also employed in all distribution-quality video coding formats, since it is able to achieve the smallest prediction error, which is then easier to code. Motion can be characterized in terms of either a velocity vector \mathbf{v} or displacement vector \mathbf{d} and is used to warp a *reference frame* onto a *target frame*. Motion estimation is used to obtain these displacements, one for each pixel in the target frame.

Several methods of motion estimation are commonly used:

- Block matching
- Hierarchical block matching
- Pel-recursive motion estimation
- Direct optical flow methods
- Mesh-matching methods

Optical flow is the apparent displacement vector field $\mathbf{d} = (d_1, d_2)$ we get from setting (i.e., forcing) equality in the so-called *constraint equation*

$$x(n_1, n_2, n) = x(n_1 - d_1, n_2 - d_2, n - 1). \quad (11.2-1)$$

All five approaches start from this basic equation, which is really just an idealization. Departures from the ideal are caused by the covering and uncovering of objects in the viewed scene, lighting variation both in time and across the objects in the scene, movement toward or away from the camera, as well as rotation about an axis (i.e., 3-D motion). Often the constraint equation is only solved approximately in the least-squares sense. Also, the displacement is not expected to be an integer as assumed in (11.2-1), often necessitating some type of interpolation to be used.

Motion cannot be determined on a pixel-by-pixel basis since there are two components for motion per pixel, and hence twice the number of unknowns as equations. A common approach then is to assume the motion is constant over a small region called the *aperture*. If the aperture is too large, then we will miss detailed motion and only get an average measure of the movement of objects in our scene. If the aperture is too small, the motion estimate may be poor to very wrong. In fact, the so-called *aperture problem* concerns the motion estimate in the square region shown in Figure 11.2-1.

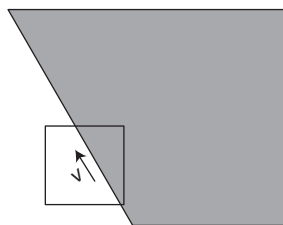
**FIGURE 11.2-1**

Illustration of the *aperture problem* with the square indicating the aperture size.

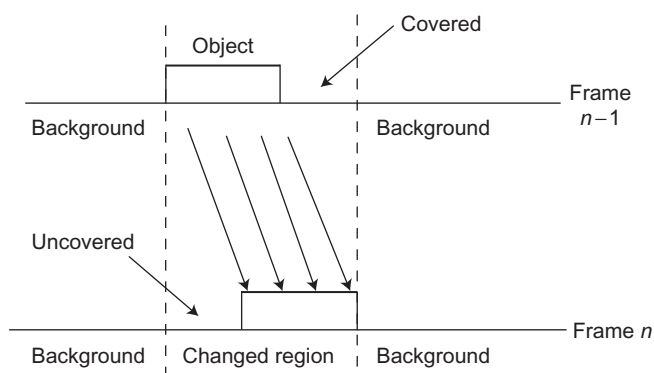
**FIGURE 11.2-2**

Illustration of covering and uncovering of background by an object moving in the foreground.

If the motion of the uniform dark region is parallel to its edge, then this motion cannot be detected. Since this situation would typically only hold for small regions in natural images, the aperture effect leads us to choose a not-too-small aperture size. Thus, finding the right aperture size is an important problem that depends on the video content.

Another issue is *covering* and *uncovering*, as illustrated in Figure 11.2-2, showing a 1-D depiction of two successive frames n and $n - 1$, with an object moving to the right. We assume a simple object translating in the foreground over a fixed background, not an unreasonable local approximation of video frames. We see that part of the background region in target frame n is uncovered, while part of the background region in reference frame $n - 1$ is covered. Motion estimation that tries to match regions in the two frames will not be able to find good matches in either the covered or uncovered regions. However, within the other background regions, matches should be good and matching should also be good within a textured object, at least if

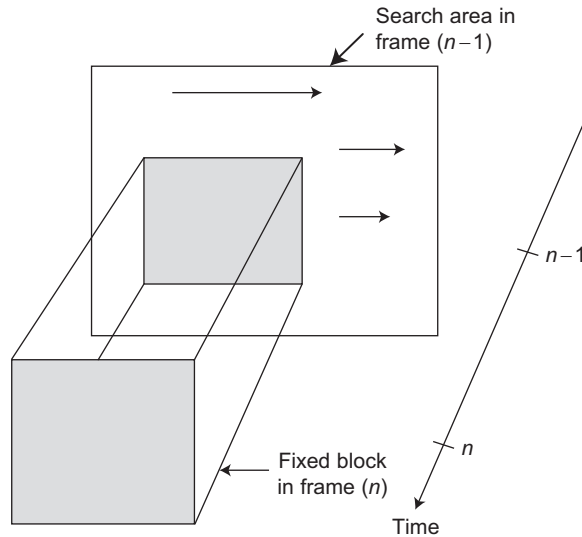
**FIGURE 11.2–3**

Illustration of simple block matching.

it moves in a trackable way, and the pixel samples are dense enough. The problem in the relatively small covered/uncovered regions is that there are two motions present there.

Block-Matching Method

We intend to estimate a displacement vector at the location (n_1, n_2, n) in the target frame. In *block matching* (BM) [5], we use template matching of the block centered on this point to blocks in a specified *search area* in the reference frame, as illustrated in Figure 11.2–3, where we take the immediately prior frame as reference.

Often the search area size is given as $(\pm M_1, \pm M_2)$ and centered on the current pixel location (n_1, n_2) in the reference frame, then a total of $(2M_1 + 1)(2M_2 + 1)$ searches must be done in a *full search*, and this must be done for each pixel where the motion is desired. Often the block matching is not conducted at every pixel in the target frame and an interpolation method is used to estimate the motion in between these points. Common error criteria are *mean-square error* (MSE), *mean-absolute error* (MAE),² or even number of pixels in the block actually disagreeing for discrete-amplitude or digital data.

²Actually preferred in practical applications because MAE works a bit better than MSE owing to being more robust, and it only involves additions. MAE goes by other acronyms, too: mean absolute difference (MAD) and sum absolute difference (SAD).

We express the MSE in block matching as

$$\mathcal{E}(\mathbf{d}) \triangleq \sum_k (x(\mathbf{n} + \mathbf{k}, n) - x(\mathbf{n} + \mathbf{k} - \mathbf{d}, n - 1))^2, \quad (11.2-2)$$

for a square block centered at position $\mathbf{n} = (n_1, n_2)^T$ as a function of the vector displacement $\mathbf{d} = (d_1, d_2)^T$. We seek the displacement vector that minimizes this error

$$\mathbf{d}_o \triangleq \arg \min_{\mathbf{d}} \mathcal{E}(\mathbf{d}).$$

Since the MSE in (11.2-2) is susceptible to outliers, often the MAE is used in applications. In addition to its being less sensitive to statistical outliers, another advantage of MAE is that it is simpler to compute.

In addition to the computationally demanding full-search method, there are simpler approximations involving a much reduced number of evaluations of the error (11.2-2). The methods either involve sampling the possible locations in the search region or sampling the elements in the block when calculating (11.2-2). Two examples of the former strategy are 2-D log search and three-step search. With reference to Figure 11.2-4, the three-step search proceeds as follows: First, the search window is broken up into four quadrants, and motion vectors are tested on a 3×3 grid with corners centered in the four quadrants. Here, we illustrate a case where the lower right corner is the best match at the first step, the top right corner is best at the second step, and the top right corner is best at the third and final step. A performance comparison is shown in Figure 11.2-5 from [6].

We note that all three techniques perform much better than simple frame differencing, which equivalently treats all displacement vectors as zero. While both fast

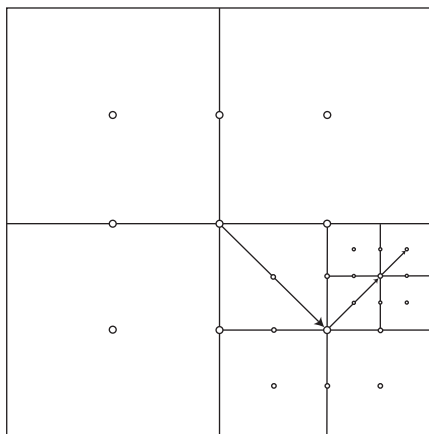


FIGURE 11.2-4

An illustration of three-step block matching.

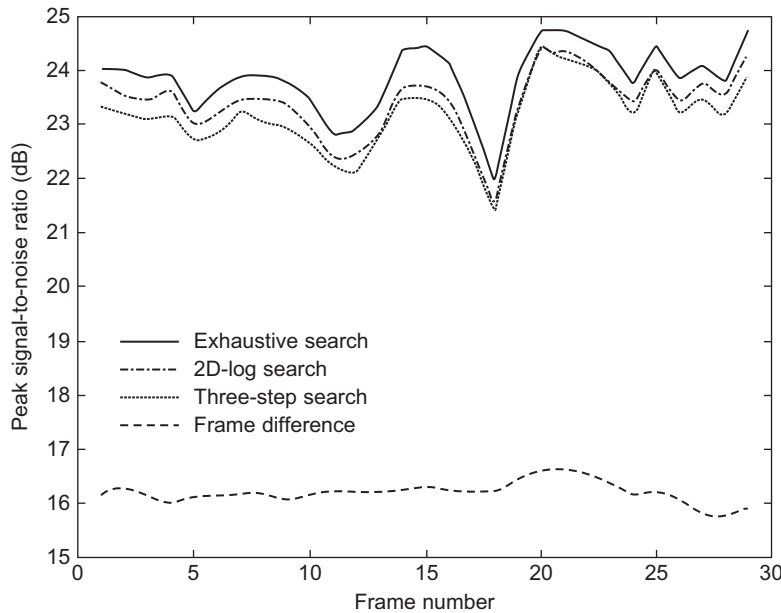
**FIGURE 11.2-5**

Illustration of PSNR performance of exhaustive, 2-D log, three-step search, and simple frame difference. ([6] ©1995 Academic Press)

methods cut computation by a factor of 10 or more versus full-search block matching, they can lose up to 1–2 dB in *prediction peak SNR* (PSNR), measured in decibels (dB) as

$$PSNR = 10 \log_{10} \left(\frac{255^2}{\mathcal{E}(d)} \right),$$

for the 8-bit images being considered, since their peak value would be 255. For 10-bit images, the formula would substitute 1023 for the peak value.

The other class of methods to speed up block matching involve sampling in the calculation of the error (11.2-2). So the amount of computation in evaluating the distortion for each searched location is reduced. Liu and Zaccarin [7] presented a method involving four phases of subsampling and alternated subsampling patterns among the blocks, while using the MAE error criterion. This method achieved approximately a four-fold reduction in the amount of computation, but only a slight increase in the average prediction MSE.

There are some unavoidable problems with the block-matching approach. A small block size can track small moving objects, but the resulting displacement estimate is

then sensitive to image noise.³ For example, a small block might just span a flat region in the image, where the displacement cannot be defined. A large block size is less sensitive to noise, but cannot track the motion of small objects. Similarly, a large search area can track fast motion but is computationally intensive. A small search area may not be large enough to catch or track the real motion.

The best matching block is often good enough for a block-based predictive video compression, where bits can be spent coding the prediction residual. However, in video filtering, when the estimated motion is not the true physical motion, visible artifacts will often be created. Hence, we need an improvement on the basic block-matching method for the filtering application. The same is true for MC interpolation, frame-rate conversion, and pre- and postprocessing in video compression. Also, some highly scalable video coders use MC in a temporal filtering structure to generate lower frame-rate versions of the original video. Highly accurate motion vectors are important in this case too. A variation of block matching, called hierarchical block matching, can achieve a much better estimate of the “true motion.”⁴

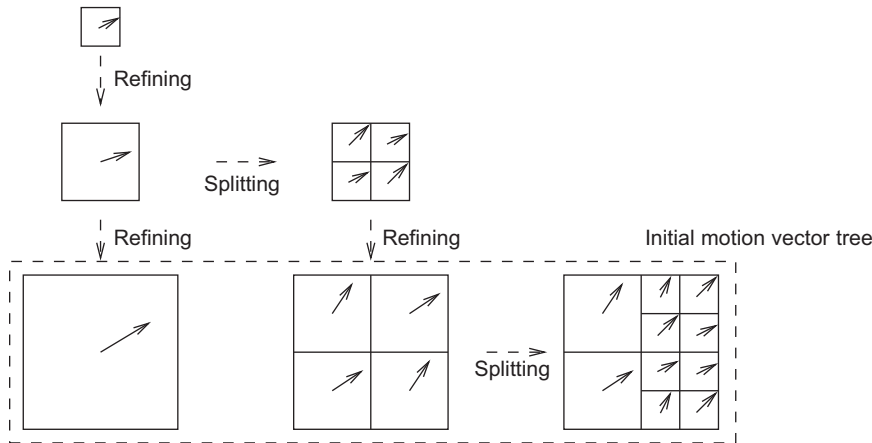
Hierarchical Block Matching

The basic idea of hierarchical algorithms is to first estimate a coarse motion vector at low spatial resolution. Then this estimated motion is refined by increasingly introducing higher spatial frequencies. Both subband/wavelet pyramids and Gaussian pyramids have been used for this purpose. An often cited early reference on *hierarchical block matching* (HBM) is Thoma and Bierling [8].

We start by creating a spatial pyramid, with resolution levels set at a power of two. Typically three or four stages of resolution are employed. We start at the lowest resolution level (highest pyramid level) and perform simple block matching. It has been found helpful to have the block size there agree with the average size of the largest moving areas in the video. Next, we start down the pyramid, increasing the resolution by the factor 2×2 at each level. We double the displacement vector from the previous level to get the initial search location at the present level. We finish up at the pyramid base level, which is full resolution. Both the block sizes and the search regions can be chosen distinctly for each resolution and are generally in the range from 4×4 to 32×32 . The maximum search area is usually small at each resolution (i.e., ± 2), since only refinements are needed. The search area may also be small at the initial pyramid level because of the low resolution. Thus we can expect considerable savings in complexity for HBM. Some other improvements to block matching include subpixel accuracy, *variable-size block matching* (VSBM), overlapping blocks, block prediction of motion vectors, and *hierarchical VSBM* (HVSBM).

³By the way, it turns out that some amount of noise is always present in real images. This comes from the common practice of setting the bit depth on sensors and scanners to reach the first one or two bits of the physical noise level, caused by photons, film grains, etc.

⁴Really it is optical flow—i.e., the apparent movement that comes from our 2-D observations.

**FIGURE 11.2-6**

An illustration of the refining and splitting process in HVSBM.

A diagram of HVSBM is shown in Figure 11.2-6. We start with a spatial pyramid that can be obtained as a succession of LL subbands by subband/wavelet filtering and 2×2 decimation. Starting at the coarsest level, at the top of the pyramid, a block-matching motion estimation is performed. Then this displacement estimate \mathbf{d}_0 is propagated down the pyramid one level, and $2\mathbf{d}_0$ is used to initialize a search over a small region to *refine* the motion value to \mathbf{d}_1 . At this time, if the MC error measure is too large, the block is *split* into four, and the process of refining is repeated to generate \mathbf{d}_1 , and this process of refining and splitting is carried down the pyramid to the bottom, resulting in a variable size block-based motion field. In a computationally more demanding variation of HVSBM, we start at the coarsest resolution with the smallest block size, and refine this motion field to the bottom of the pyramid (i.e., the highest resolution level). Then this resulting motion field is *pruned* back by merging nodes to a variable-size block-based motion field. This can be done using the BFOS algorithm, and this *bottom-up* approach generally results in a more accurate motion field than the top-down method mentioned in the previous paragraph, but it is more computationally intensive.

In a video coding application, the error criteria can be composed of motion field error, either MSE or MAE, to weigh the increase in motion-vector rate due to the split (top-down) or decrease in motion-vector rate due to the merge (bottom-up). A Lagrangian approach is often used to control the bitrate of the motion information. More on coding motion vectors for video compression is contained in Chapter 12.

Overlapped Block Motion Compensation

The motivation to overlap the blocks used in a conventional block-matching estimate is to increase the smoothness of the resulting velocity field. This can be considered as

a method to reduce the spatial frequency aliasing in sampling the underlying velocity field. While one could simply overlap the blocks used in a simple block-matching estimate, this could mean much more computation. For example, if the blocks were overlapped by 50% horizontally and vertically, it would be four times more computation if the block-matching estimation were done independently, as well as four times more velocity information to transmit in the video compression application of Chapter 12. So, effectively we are more interested in smoothing than in alias reduction, and the *overlapped block motion compensation* (OBMC) technique [9, 10] simply weights each velocity vector with four neighbor velocity estimates from the four nearest neighbor nonoverlapping blocks. Thus we effectively overlap the velocity vectors without overlapping the blocks themselves. This is usually done with a few prescribed weighting windows.

A theoretical motivation for the overlapping can be obtained from (4) of [10],

$$\begin{aligned}\hat{x}(\mathbf{n}, n) &= E\{x(\mathbf{n}, n) | \mathbf{X}(n-1), \mathcal{V}_{\mathbf{n}}\} \\ &= \int f_{\mathbf{n}}(\mathbf{v} | \mathcal{V}_{\mathbf{n}}) x(\mathbf{n} - \mathbf{v}\Delta t, n-1) d\mathbf{v},\end{aligned}$$

only slightly changed for our notation. Here, we are performing a motion-compensated estimate of frame $\mathbf{X}(n)$, as a conditional mean over shifted versions of frame $\mathbf{X}(n-1)$, with interframe interval Δt , making use of the conditional pdf $f_{\mathbf{n}}(\mathbf{v} | \mathcal{V}_{\mathbf{n}})$, which depends on $\mathcal{V}_{\mathbf{n}}$, the motion vector data sent in this and neighboring blocks. Assuming linear weights (i.e., no dependence on the data values in $\mathcal{V}_{\mathbf{n}}$), they obtain

$$\hat{x}_n(\mathbf{n}) = \sum_{N_b(\mathbf{x})} w_b(\mathbf{n}) x(\mathbf{n} - \mathbf{v}_b \Delta t, n-1), \quad (11.2-3)$$

where the sum is over velocity vectors in the neighboring blocks $N_b(\mathbf{n})$. A formula for obtaining an optimized block weighting function $w_b(\mathbf{n})$ is given in [10]. Simple weighting windows are given there too.

The initial estimate obtained in this way can be improved upon by iteratively updating the velocity estimates from the various blocks, one at a time, by utilizing the resulting overlapped estimate (11.2-3) in the error calculation (11.2-2). OBMC is used in the H.263 video compression standard for visual conversation and has also been adapted for use in some SWT video coders. In the compression application, it can smooth the velocity field without the need to transmit additional motion vector bits, since the block overlapping can be done separately at the receiver given the transmitted motion vectors, still only one for each block. The overlapping of the blocks makes the velocity field smoother and removes the artificial blocked structure. This is especially important for SWT coders, where a blocky motion vector field could lead, through motion compensation, to a blocky prediction residual that would have false and excessively high spatial frequency information.

Pel-Recursive Motion Estimation

This iterative method recursively calculates a displacement vector for each pixel (a.k.a. pel) in the current frame. We start with an estimate $\mathbf{d} = (d_1, d_2)^T$ for the current displacement. Then we use the iterative method,

$$\begin{aligned}\hat{d}_1^{(k+1)} &= \hat{d}_1^{(k)} - \epsilon \frac{\partial \mathcal{E}}{\partial d_1} \Big|_{\mathbf{d}=\hat{\mathbf{d}}^{(k)}}, \\ \hat{d}_2^{(k+1)} &= \hat{d}_2^{(k)} - \epsilon \frac{\partial \mathcal{E}}{\partial d_2} \Big|_{\mathbf{d}=\hat{\mathbf{d}}^{(k)}},\end{aligned}$$

with initial value supplied by the final value at the previously scanned pixel,

$$\hat{\mathbf{d}}^{(0)}(n_1, n_2) = \hat{\mathbf{d}}^{(\text{final})}(n_1 - 1, n_2).$$

A key reference is [11]; see also [6]. The method works well with just a few iterations when the motion is small, but often fails to converge when the displacements are large. In [12], this differential displacement approach was extended to hierarchically structured motion estimation, with application to image sequence frame interpolation.

Optical Flow Methods

Optical flow is a differential method that works by approximating the derivatives rather than the function error itself, as in block matching. It is a least-squares approximation to the *spatiotemporal constraint equation*,

$$v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} = 0, \quad (11.2-4)$$

which is derived by partial differentiation of the optical flow equation (11.2-1), rewritten as a function of real variables (x, y) with velocity parameters v_x and v_y ,

$$f(x, y, t) = f(x - v_x dx, y - v_y dy, t - \Delta t).$$

Because of noise in the frame, (11.2-4) is then subjected to least squares approximation to give the optical flow velocity estimate. Specifically, we form the error criteria

$$\mathcal{E}_{MV} \triangleq \left(v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} \right)^2,$$

to be minimized over local regions.

In practice, a smoothing term must be added to this error term to *regularize* the estimate, which otherwise would be much too rough—i.e., too much high-frequency energy in the estimate $\hat{\mathbf{v}}(x, y, t)$. In the Horn and Schunck method [13], a gradient smoothness term is introduced via a Lagrange multiplier as

$$\begin{aligned}\lambda \mathcal{E}_S &\triangleq \lambda \left[\|\nabla v_x\|^2 + \|\nabla v_y\|^2 \right] \\ &= \lambda \left[\left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 \right],\end{aligned}$$

which, for large values of the positive parameter λ , makes the velocity estimate change slowly as a function of the spatial variables x and y .

Integrating over the area of the image, we get the total error to be minimized as

$$\begin{aligned}\mathcal{E}_T &= \int \int (\mathcal{E}_{MV} + \lambda \mathcal{E}_S) dx dy \\ &= \int \int \left\{ \left(v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} \right)^2 + \lambda \left[\left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 \right. \right. \\ &\quad \left. \left. + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 \right] \right\} dx dy.\end{aligned}$$

We seek the minimizing velocity vector

$$\hat{\mathbf{v}} \triangleq \arg \min \mathcal{E}_T(\mathbf{v}).$$

The calculus of variations is then used to find the minimum of this integral in terms of the unknown functions $v_x(x, y, t)$ and $v_y(x, y, t)$ for each fixed frame t . The resulting equations are then approximated using first-order approximations for the various derivatives involved. Longer digital filters may provide improved estimates of these derivatives of the, assumed bandlimited, analog image frames [14]. An iterative solution is then obtained using Gauss-Seidel iterations.

While this estimate has been used extensively in computer vision, it is not often used in video compression because of its rather dense velocity estimate. However, optical flow estimates have been used extensively in video filtering, where the need to transmit the resulting motion vectors does not occur. There it can give a smooth and consistent performance, with few motion artifacts. A modern optical flow method is presented in Section 5.4 of Chapter 5 in *The Essential Guide to Video* [15]. The main problem with optical flow methods is that the smoothness of their motion does not allow discontinuities of motion across object boundaries in the scene.

Mesh-Based Methods

In a mesh-based method, similar to block-based, a regular grid of velocity points called *control points* is set up in the target frame and the corresponding matched points in a reference frame. But unlike block matching, the motion is not considered constant between the control points. Rather, these points are used to set up an *affine motion model*. An affine model has six parameters and represents rotation and translation, projected onto the image plane, as

$$\begin{aligned}d_1(x_1, x_2) &= a_{11}x_1 + a_{12}x_2 + a_{13}, \\ d_2(x_1, x_2) &= a_{21}x_1 + a_{22}x_2 + a_{23},\end{aligned}$$

where the position vector $\mathbf{x} = (x_1, x_2)^T$ and $\mathbf{d}(\mathbf{x})$ is the displacement vector. We can see translational motion as the special case where only a_{13} and a_{23} are nonzero and the displacement is constant at $(d_1, d_2) = (a_{13}, a_{23})$ in this block. The motion warping

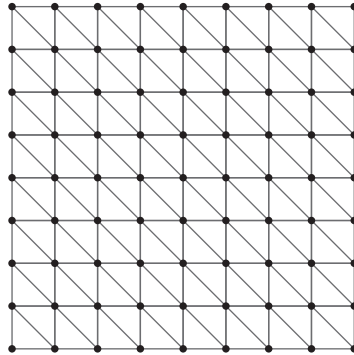
**FIGURE 11.2-7**

Illustration of regular triangular mesh grid on target frame.

effect of an affine motion model has been found to well approximate the apparent motion or optical flow of pixels on rigid objects. If we break up the squares of the regular grid in the target frame into triangles, we get triangular patches, with three control points at the vertices of each triangle, as seen in Figure 11.2-7, and a separate affine model can be determined for each triangle patch—i.e., six linear equations in the six unknowns ($a_{11}, a_{21}, a_{12}, a_{22}, a_{13}, a_{23}$).

Because the control points are shared between two adjoining patches, the resulting velocity field will be continuous across the patch boundaries unlike the case in block matching. In the reference frame, when the control points are properly matched, the triangular grid will appear warped. This grid shape indicates that spatially warped prediction is being applied from the reference frame to the target frame. The continuity of the velocity field makes the prediction more subjectively acceptable but does not usually lead to better objective error performance in either an MSE or MAE sense. Basically, a small geometric error is hardly noticeable, but it can affect the objective error a lot. Pseudo code for performing mesh matching follows:

```

FOR each grid point
    Do block matching, with block centered by grid point, to find  $d_i$ .
     $T_i = 1$ ;
END FOR
WHILE not exceed maximum iterations
    FOR each grid point  $V_i$ 
        IF  $T_i == 1$ 
            Refine the motion vector by  $\min_{d_i} \sum_k E_k$ , where  $E_k$  is
            the prediction error of each triangle that connected
            to this grid point
        IF  $d_i$  does not change
             $T_i = 0$ ;
        ELSE
             $T_i = 1$ ;
    
```

```

        FOR any grid point  $V_j$  that edge-connects with  $V_i$ 
           $T_j = 1$ ;
        END FOR
      END FOR
    IF  $\sum T_i < 1$  BREAK
  END WHILE

```

This algorithm first performs block matching to get initial displacement estimates for the control grid points. It then iteratively visits all the grid points in succession, looking for better affine model fits. The T_i variable keeps track of whether the control vector at that grid point is converged or not. The following example has used this algorithm with the MAE error criteria to perform a mesh-based motion estimation. The max number of iterations was set to three, and the search area was set at $(\pm 31, \pm 31)$.

Example 11.2–1: Mesh Matching Versus Block Matching

We look at two examples of warping frame 93 of the CIF⁵ clip *foreman* at 30 fps to match the next frame 94. There is enough motion between these two frames to illustrate the shortcomings of each approach. We use fixed-size 32×32 blocks here. Figure 11.2–8 shows frame 94 with the fixed-size triangular grid overlaid upon it. Figure 11.2–9 shows frame 93 with the found warped triangular mesh overlaid. We see that there is considerable movement between these two frames, and it is mainly in the foreman’s mouth and chin region. Figure 11.2–10 shows the resulting spatially warped estimate of frame 94. It clearly displays warping errors, most obvious in the lower facial region. Finally, in



FIGURE 11.2–8

Frame 94 of the foreman clip with fixed-size triangular grid overlaid.

⁵See appendix on video formats at the end of this chapter.

**FIGURE 11.2-9**

Frame 93 of the foreman clip with warped grid overlaid.

**FIGURE 11.2-10**

Warped prediction of frame 94 of the foreman clip from the preceding frame using triangular fixed-size mesh matching.

Figure 11.2-11 we show the corresponding block-matching estimate for the same 32×32 fixed size grid. We can see obvious blocking artifacts here. We see evident distortions in each prediction of frame 94, mostly near the mouth region of the foreman's face. You can watch the full videos included in a folder on this book's Web site. ■

The mesh may be fixed size or variable size, and the motion parameters a_{ij} may be estimated hierarchically or not. A popular choice is *variable-size mesh matching* (VSMM). Use of variable size blocks can reduce both the blocking and warping artifacts in this example. Finally, and distinct from mesh matching, there also exist generalized block models with affine or more general polynomial motion models within each block [16].

**FIGURE 11.2–11**

Fixed-size block-matching prediction of frame 94 of the foreman clip from the preceding frame.

11.3 MOTION-COMPENSATED FILTERING

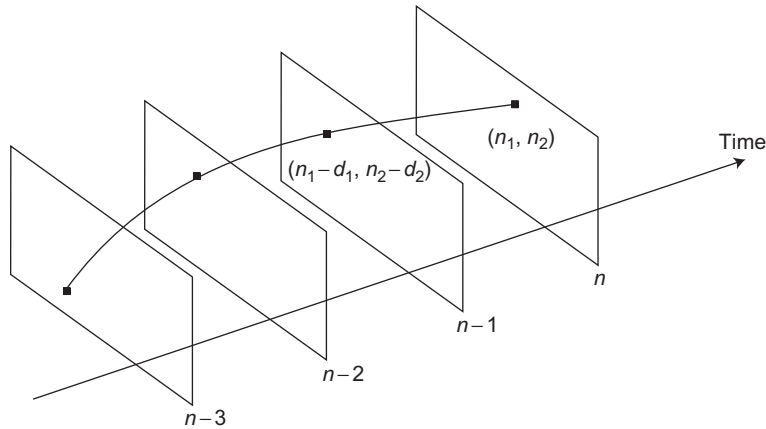
If the motion is slow or there is no motion at all, then simple temporal filtering can be very effective for estimation, restoration, frame-rate change, interpolation, or pre- or postprocessing. In the presence of strong motion, however, artifacts begin to appear in the simple temporal filter outputs and the needed coherence in the input signal begins to break down. Such coherence is needed to distinguish the signal from the noise, distortion, interference, and artifacts that may also be present. A solution is to modify the filter trajectory so that it follows along the trajectory of motion. In this way, signal coherence is maintained, even with moderate to fast motion.

The basic idea is to modify an LSI filter as follows:

$$\begin{aligned}
 y(n_1, n_2, n) = & \sum_{k_1, k_2} h(k_1, k_2, 0) x(n_1 - k_1, n_2 - k_2, n) \\
 & + \sum_{k_1, k_2} h(k_1, k_2, 1) x(n_1 - d_1 - k_1, n_2 - d_2 - k_2, n - 1) \quad (11.3-1) \\
 & + \text{etc.}
 \end{aligned}$$

Here, $d_1 = d_1(n_1, n_2, n)$ is the horizontal component of the displacement vector between frames n and $n - 1$, and d_2 is the vertical component of displacement. In order to get the corresponding terms for frame $n - 2$, we must add the displacement vectors from the frame pair $n - 1$ and $n - 2$ to get the correct displacement. We should add them vectorially:

$$\begin{aligned}
 d'_1(n_1, n_2) &= d_1(n_1, n_2, n) + d_1(n_1 - d_1(n_1, n_2, n), n_2 - d_2(n_1, n_2, n), n - 1), \\
 d'_2(n_1, n_2) &= d_2(n_1, n_2, n) + d_2(n_1 - d_1(n_1, n_2, n), n_2 - d_2(n_1, n_2, n), n - 1).
 \end{aligned}$$

**FIGURE 11.3-1**

An illustration of motion-compensated filtering along a motion path.

Here, we assume that the displacement vectors are known, most likely because they were estimated previously, and that they are integer valued. If they are not integer valued, which is most of the time, then the corresponding signal value, such as $x(n_1 - d_1 - k_1, n_2 - d_2 - k_2, n - 1)$, must itself be estimated via interpolation. Most often, spatial interpolation is used based on the use of various lowpass filters. Effectively, in (11.3-1) we are filtering along the motion paths rather than simply filtering straight forward (or backward) in time (n) at each spatial location. One way to conceptualize this is via the diagram in Figure 11.3-1.

MC-Wiener Filter

In Figure 11.3-2, MC denotes *motion-compensated warping* performed on the noisy observations $y(n_1, n_2, n)$. The Wiener filtering is then done on the warped data in the MC domain, with signal and noise PSDs calculated from some similar MC data. Finally, the *inverse motion compensation* (IMC) operator dewarps the frames back to original shape to produce the output estimate $\hat{x}(n_1, n_2, n)$. Three-dimensional MC-Wiener filtering was introduced in [17]. The concept of IMC depends on the motion field being one-to-one. In a real image sequence, there is a relatively small number of pixels where this is not true due to coverings and uncoverings of objects in the scene, the so-called *occlusion problem*. In these areas, some approximation must be used in Figure 11.3-2 in order to avoid introducing artifacts into the final video estimate. It is common to resort to intraframe filtering in these occluded areas.

Because of the strong correlation in the temporal direction in most video, there is often a lot to gain by processing the video jointly in both the temporal and spatial directions. Since a Wiener filter is usually implemented as an FIR filter, exploiting the

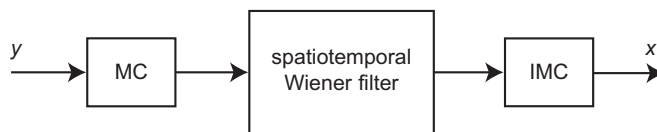
**FIGURE 11.3–2**

Illustration of MC warping followed by Wiener filter followed by IMC warping (IMC).

temporal direction in this way means that several to many frames must be kept in active memory. An alternative to this method is the spatiotemporal Kalman filter, which can use just one frame of memory to perform its recursive estimate, a motion-compensated version of which is presented next. Of course, both methods require the estimation of a suitable image sequence model and noise model. The signal model spectra can be obtained via estimation on similar noise-free data for the Wiener filter, while the Kalman filter needs parameter estimation of an autoregressive model. Both models must be trained or estimated on data that have been warped by the motion compensator, since this warped domain is where their estimate is performed.

MC-Kalman Filter

The basic idea here is that we can apply the totally ordered temporal 3-D RUKF of Chapter 10 along the motion trajectory using a good motion estimator that approximates true motion. As before, we can use multiple models for both motion and image estimation. To reduce object blurring and sometimes even double images, we effectively shift the temporal axis of the filter to be aligned with motion trajectories. When a moderate-sized moving object is so aligned, we can then apply the filter along the object's trajectory of motion by filtering the MC video. Since the MC video has a strong temporal correlation, its image sequence model will have a small prediction error variance. This suggests that high spatial frequencies can be retained even at low input SNRs via motion-compensated Kalman filtering. The overall block diagram of a motion compensated 3-D Kalman filter of Woods and Kim [1], or MC-RUKF, is shown in Figure 11.3–3.

This motion-compensated spatiotemporal filter consists of three major parts: the motion estimator, the motion compensator, and the 3-D RUKF. While filtering a video, two different previous frames could be used for motion estimation; one could use either the previous smoothed frame $E\{\mathbf{x}(n-1)|\mathbf{y}(n), \mathbf{y}(n-1), \dots\}$ or the previous noisy frame $\mathbf{y}(n-1)$. In our work, we have generally found it best to use the smoothed previous frame, since it is the best estimate currently available. For motion estimation, we used an HBM method.

The motion estimate is used to align a set of local frames along the motion trajectory. To effect this local alignment, the smoothed previous frame estimate is displaced to align with the current frame. In an iterative method extension shown in Figure 11.3–3, two smoothed frames are used to improve on the initial motion estimates. These smoothed frames retain spatial high frequencies and have reduced

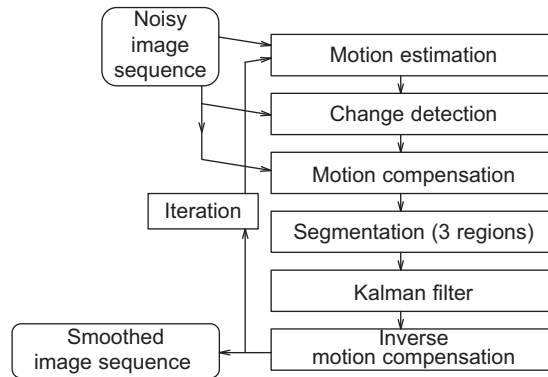


FIGURE 11.3–3

System diagram for motion-compensated spatiotemporal Kalman filter.

noise, so that these frames can now be used for motion estimation with a smaller size block. A motion vector field is then estimated from these two frames (i.e., $E\{x(n-1)|y(n), y(n-1), \dots\}$ and $E\{x(n)|y(n+1), y(n), y(n-1), \dots\}$), followed by a second application of the steady-state 3-D RUKF. A few iterations suffice.

Multimodel MC-RUKF

The local correlation between two MC frames depends on the accuracy of the motion estimation. Since the SNR of the noisy observed video will be low, the motion estimation has a further limitation on its accuracy, due to statistical variations. To deal with the resulting limited motion vector accuracy, we can use a variable number of models to match the various motion regions in the image; for example, we can use three motion models: *still*, *predictable*, and *unpredictable*. The motivation here is that in the still region, we can perform unlimited temporal smoothing at each pixel location. In the predictable region, there is motion, but it is motion that can be tracked well by our motion estimator. Here, we can smooth along the found motion trajectory with confidence. Finally, in the unpredictable region, we find that our motion estimate is unreliable and so fall back on the spatial RUKF there. This *multiple model* version (MM MC-RUKF) results in a very high temporal coherence in the still region, high temporal coherence in the predictable region, and no motion blurring in the unpredictable region. The segmentation is based on local variance of the *displaced frame difference* (DFD).

As mentioned earlier, we employ a block-matching method for motion estimation. Even when there is no correspondence between two motion-compensated frames, the block-matching method chooses a pixel in the search area that minimizes the displaced frame difference measure. However, the estimate will probably not have much to do with the real motion, and this can lead to low temporal correlation in the unpredictable region. This is the so-called *noisy motion vectors* problem. We can compensate for this, in the case of still regions, by detecting them with an extra step,

based on the frame difference. We filter the frame difference and use a simple 7×7 box filter to reduce the effect of the observation noise. Also, a 3×3 box filter is used on the MC output to detect the predictable region. The outputs are then fed into local variance detectors. We found that when a pixel in a still region was missed, a visual error in the filtered image sequence was noticeable, while in the opposite case, the error was not noticeable. Hence, we detect the still region again in the filtering step. Three spatiotemporal AR models are obtained from the residual video of the original sequence for our simulation. For more details, see [1].

Example 11.3–1: MM MC-RUKF Experimental Result

We used the CIF video salesman, which is monochrome and of size 360×280 pixels at 15 fps. We then added white Gaussian noise to achieve a 10-dB SNR. The processing parameters of the MC-RUKF were as follows: image model order $1 \times 1 \times 1$, update region $2 \times 2 \times 1$, and final MC block sizes of both 9×9 and 5×5 . The 3-D AR model obtained from the original (modified) video was used. This model could also have been obtained from the noisy video or from a prototype noise-free, with some loss of performance. (Based on existing work in the identification of 2-D image models, it is our feeling that the additional loss would not be great.) We used a steady-state gain array, calculated off-line on a small fictitious image sequence. The SNR improvement achieved was 6–8 dB with the 3-D RUKF alone, with an additional MC-RUKF improvement of about 1 dB. Using the multimodel feature, a further MM MC-RUKF improvement of about 1 dB was achieved, totaling to an 8- to 10-dB improvement or an output SNR of 18–20 dB.

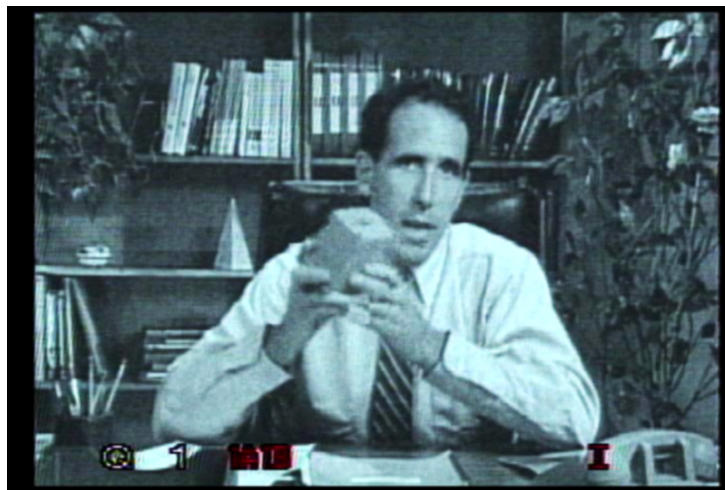


FIGURE 11.3–4

A frame from the MC-RUKF.

The restored video in Figures 11.3-4 and 11.3-5 showed motion artifacts visible in some motion areas but was generally quite visually pleasing.

The resulting SNR improvement curves are given in Figure 11.3-6. We notice that the MM MC-RUKF provides the best objective performance by this MSE-based measure. We can see an initial start-up transient of about 10 frames. We notice also how the up

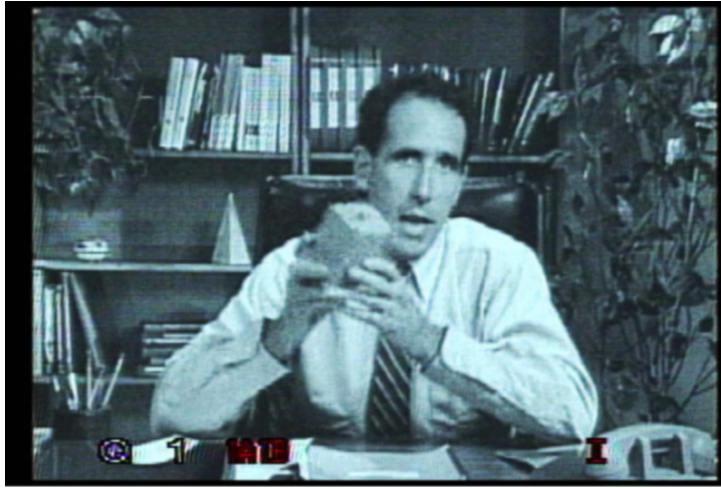


FIGURE 11.3-5

A frame from the MM MC-RUKF.

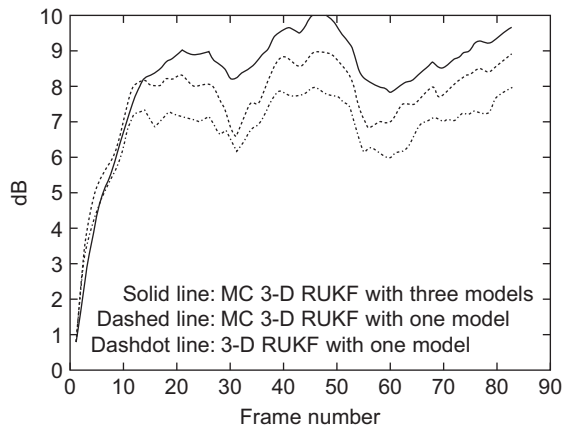


FIGURE 11.3-6

Plot of SNR improvements versus frame number for the MM MC-RUKF (3 models), MC-RUKF, and 3-D RUKF on the noisy salesman clip.

to 10-dB improvement varies across the frame number; this is caused by the motion of objects and moving shadows in the scene. Videos are available for download at the book's Web site. ■

Frame-Rate Conversion

Frame-rate conversion is needed today due to the coexistence of multiple standard frame rates (60, 30, 25, and 24 fps), and also leads to a convenient separation of acquisition format, transmission standard, and viewing or display format. Frame-rate up-conversion is also often used to double (or quadruple) the frame rate for display (e.g., 25 fps to 50 or 100 fps). There are various methods for increasing the frame rate, the simplest being frame repeat (i.e., sample and hold-in time). Somewhat more complicated is making use of a straight temporal average, without motion compensation. The filtering is a 1-D interpolation (i.e., linear filtering) in the temporal direction, done for each pixel separately. A potentially more accurate method for frame-rate increase is MC-based frame interpolation, first suggested in [8].

Example 11.3–2: Frame-rate Up Conversion

We have applied the method of [8] to the test clip Miss America, which is color and SIF sized⁶, with 150 frames at 30 fps. To perform our simulation, we first decimated it down to 5 fps and used only this low frame rate as input. Then we used the following strategies to interpolate the missing frames (i.e., raise the frame rate back up to 30 fps): frame replication, linear averaging, and motion-compensated interpolation. As a motion estimation method, we employed HBM, with the result smoothed by a simple lowpass filter.

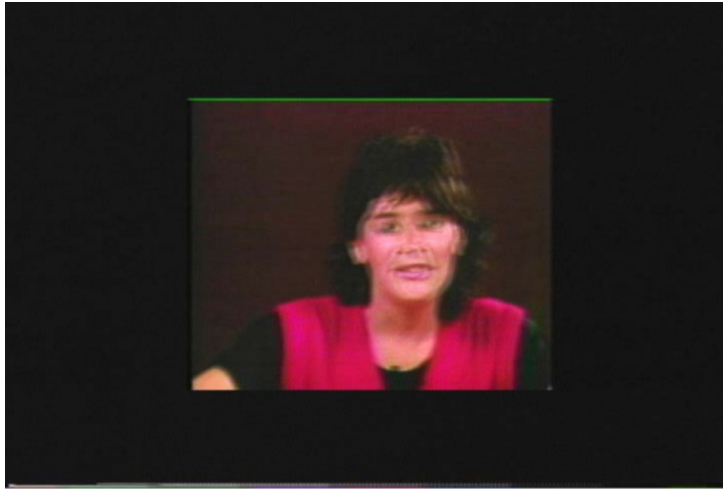
Figure 11.3–7 shows a frame from temporal up-conversion using a simple linear averaging filter. Note that during this time period, there was motion that has caused a double image effect on the up-converted frame. Figure 11.3–8 shows the result of using the motion-compensated up-conversion at a frame number near to that of the linear result in Figure 11.3–7. We do not see any double image, and the up-conversion result is generally artifact-free.

In this case our translational motion model worked very well, in part because of the rather simple motion displayed in this Miss America test clip. However, it does not always work this well, and MC up-conversion remains a challenging problem. Videos are available for download at the book's Web site. ■

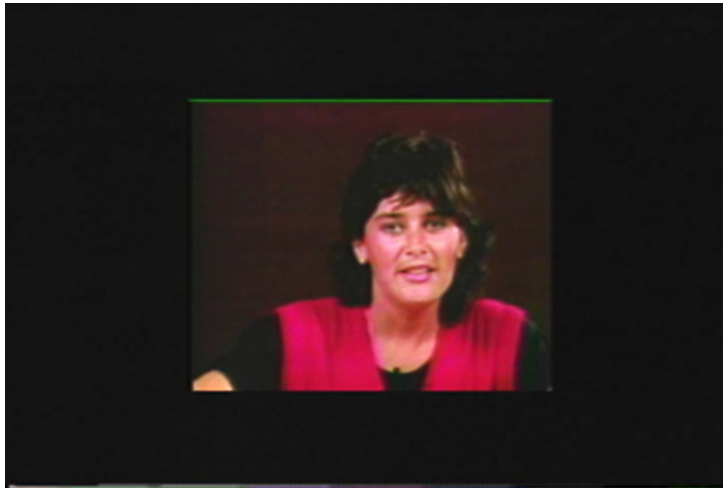
Deinterlacing

As mentioned in Example 2.2–5 of Chapter 2, deinterlacing is used to convert from a conventional interlaced format to one that is progressive or noninterlaced. In so

⁶Please see appendix on video formats.

**FIGURE 11.3-7**

A frame from a linearly interpolated temporal up-conversion of the Miss America clip from 5 to 30 fps.

**FIGURE 11.3-8**

A frame from the motion-compensated temporal up-conversion of the Miss America clip from 5 to 30 fps.

doing, the deinterlacer must estimate the missing data—i.e., the odd lines in the so-called *even frames* and the even lines in the *odd frames*. A conventional deinterlacer uses a diamond v - t multidimensional filter in upsampling the data to progressive format. If the interlaced video had been prefiltered prior to its original sampling on

this lattice to avoid spatial frequency aliasing, then using an ideal filter, the progressive reconstruction can be exact, but still with the original spatiotemporal frequency response. If proper prefiltering had not been done at the original interlaced sampling, then aliasing error may be present in both the interlaced and progressive data. In this case, a nonideal frequency response for the conversion filter can help to suppress the alias energy that usually occurs at locations of high spatiotemporal frequency. While interlaced video is not as ubiquitous as it once was, the ATSC broadcast interlaced standard 1080i is common and needs conversion to 1080p for a progressive display.

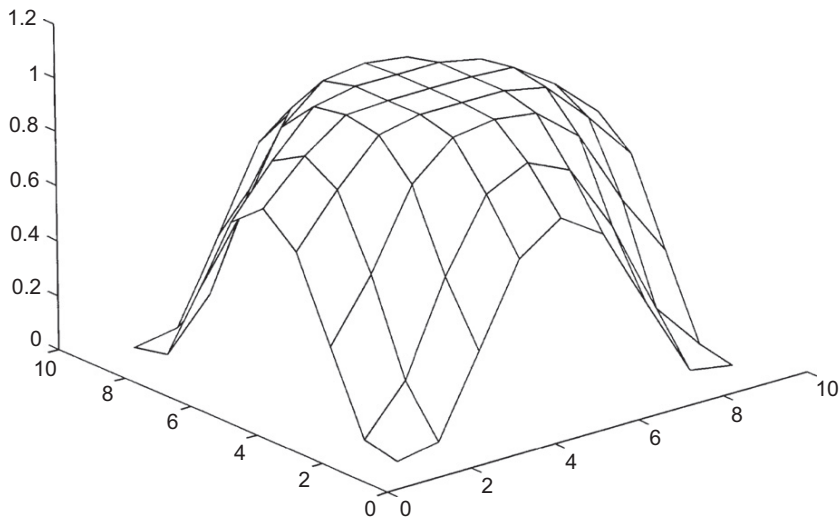
Example 11.3–3: Conventional Deinterlacer

This example uses a 9×9 diamond-shaped support in the $v \times t$ plane. The filter coefficients, shown in Table 11.3–1, were obtained via window-based FIR filter design. The frequency response of this filter is shown in Figure 11.3–9, where we can see a broader response along both temporal and vertical frequency axes than along diagonals, hence approximating a diamond pattern in the $v \times t$ ($n_2 \times n$) frequency domain.

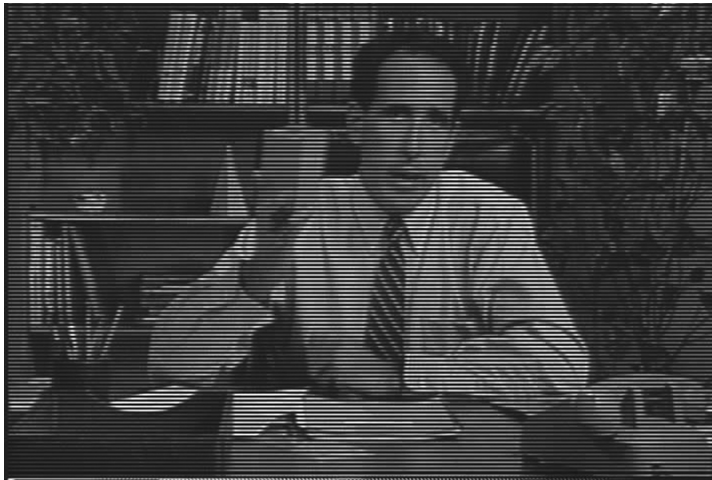
Figure 11.3–10 shows one field from the interlaced salesman sample, obtained by filtering and downsampling from the corresponding progressive clip. It serves as the starting point for our deinterlacing experiments. Figure 11.3–11 shows a frame from the resulting *progressive* (noninterlaced) output. We note that the result is generally pleasing, if somewhat soft (slightly blurred). From the frequency response in Figure 11.3–9, we can see that the image frame sharpness should be generally preserved for low temporal frequencies (i.e., slowly moving or stationary objects). Fast-moving objects, corresponding to diagonal support on the $v \times t$ filter frequency response function, will be blurred. Videos are available for download at the book’s Web site.

While blurring of fast-moving objects is generally consistent with the limitations of the human visual system response function, coherent motion can be tracked by the viewer. As such, it appears as low temporal frequency on the tracking viewer’s retina, and hence the blurring of medium- to fast-moving objects can be detected for so-called *trackable motion*.

Table 11.3–1 Diamond Filter Coefficients								
0	0	0	0	0.001247	0	0	0	0
0	0	0	0.004988	–0.005339	0.004988	0	0	0
0	0	0.007481	–0.016016	–0.013060	–0.016016	0.007481	0	0
0	0.004988	–0.016016	–0.036095	0.162371	–0.036095	–0.016016	0.004988	0
0.001247	–0.005339	–0.013060	0.162371	0.621808	0.162371	–0.013060	–0.005339	0.001247
0	0.004988	–0.016016	–0.036095	0.162371	–0.036095	–0.016016	0.004988	0
0	0	0.007481	–0.016016	–0.013060	–0.016016	0.007481	0	0
0	0	0	0.004988	–0.005339	0.004988	0	0	0
0	0	0	0	0.001247	0	0	0	0

**FIGURE 11.3-9**

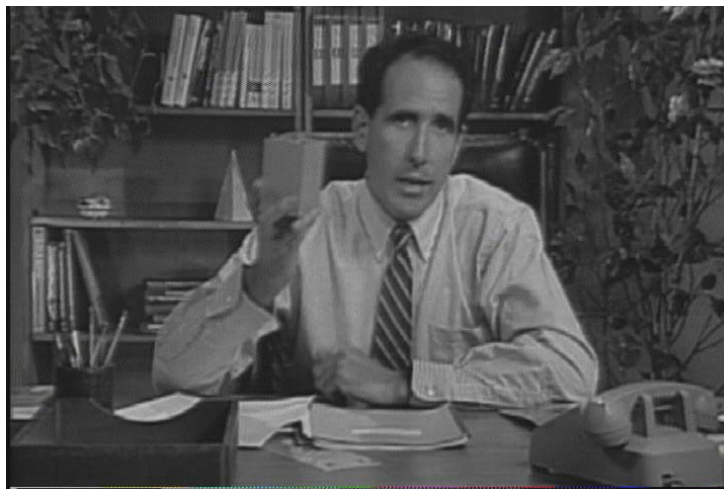
Sketch of diamond filter response in the vertical-temporal frequency domain.

**FIGURE 11.3-10**

One field from the interlaced version of the salesman clip.

Example 11.3-4: Median Deinterlacer

An alternative to the use of the classic multidimensional filter is the vertical-temporal median filter. The most common method uses a three-pixel median filter, with one pixel

**FIGURE 11.3–11**

A progressive frame from the diamond filter ($v \times t$) output for an interlaced input.

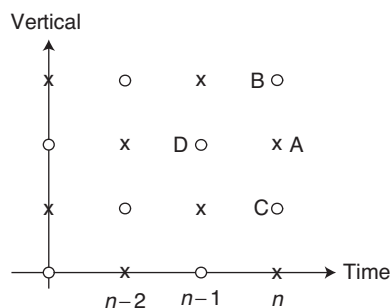
**FIGURE 11.3–12**

Illustration of pixels input (B, C, and D) to the median filter deinterlacer.

above and below the current pixel, and one pixel right behind it in the previous field. On the progressive lattice, this median operation can be written as follows:

for odd frames,

$$\hat{x}(n_1, 2n_2, n) = \text{median}\{x(n_1, 2n_2 + 1, n), x(n_1, 2n_2 - 1, n), x(n_1, 2n_2, n - 1)\},$$

for even frames,

$$\hat{x}(n_1, 2n_2 + 1, n) = \text{median}\{x(n_1, 2(n_2 + 1), n), x(n_1, 2n_2, n), x(n_1, 2n_2 + 1, n - 1)\}.$$

In [Figure 11.3–12](#), circles indicate pixels (lines) present in a field, while x's represent missing pixels (lines). We see three input pixels (B, C, and D) and one output

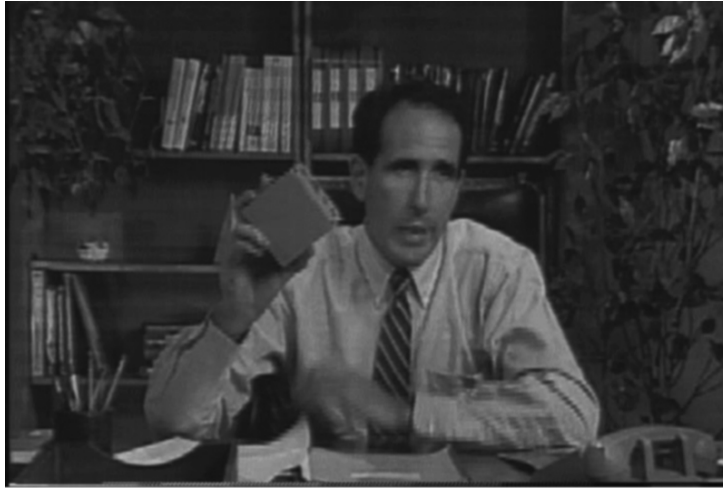


FIGURE 11.3–13

A deinterlaced frame of the salesman clip by the adaptive median filter.

pixel (A), which represent a missing pixel at field n . The statistical median of the three input values (B, C, and D) will tend to favor D if there is no motion, but to favor B or C in the case of motion. Thus this simple median deinterlacer switches back and forth between temporal and spatial (vertical) interpolation to fill in the missing pixel lines in the even and odd fields. A vertical-temporal median deinterlaced frame is shown in Figure 11.3–13. While the result is sharper than that of the multidimensional filter, this sharpness is obtained at the cost of small artifacts occurring on fast-moving objects. Videos are available for download at the book's Web site. ■

A more powerful alternative is motion-compensated deinterlacing. It uses motion estimation to find the best pixels in the previous field or fields for the prediction of the current pixel in the missing lines of the current field.

Example 11.3–5: Motion-Compensated Deinterlacer

In this example we try to detect and track motion [18] and then use it to perform the deinterlacing. Using an HBM motion estimator based on a QMF SWT, we first determine if the velocity is zero or not, based on looking at a local average of the mean-square frame difference and comparing it to a threshold. A simple three-tap vertical interpolation filter was used to deinterlace at this first stage. The motion is then said to be “trackable” if the local motion-compensated MSE is below a second threshold. The algorithm then proceeds as follows:

- When no motion is detected, we smooth in the temporal direction only (i.e., use the pixel at the same position in the previous field).

- When motion is detected, and with reference to Figure 11.3–14, we project the motion path onto the previous two fields, with a *cone of uncertainty* opening to 0.1–0.2 pixels at the just prior field. If the cone includes a pixel in the first prior field, then that pixel is copied to the missing pixel location A in the current field. Otherwise, we look to the second prior field. If no such previous pixels exist in the “cone regions,” we perform linear spatiotemporal interpolation.

The result in Figure 11.3–15 is potentially the best of these examples shown, albeit the most computationally demanding due to the motion estimation. It is sharp and clear, but unfortunately suffers from some motion artifacts, which could at least be partially ameliorated by more sophisticated motion estimation and compensation methods. The frame rate of the salesman clip was 15 fps.

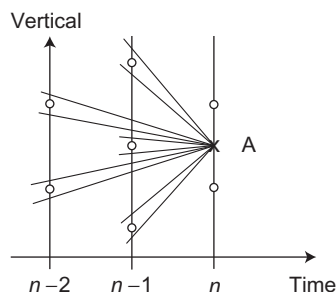


FIGURE 11.3–14

An illustration of the cone approach to motion-compensated deinterlacing.

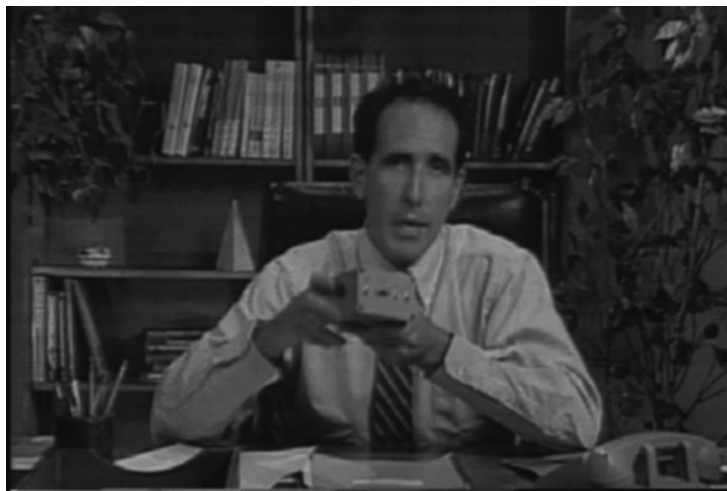


FIGURE 11.3–15

An MC deinterlaced frame from the salesman clip.

One last comment on deinterlacing: Of course, missing data not sampled cannot be recovered without some assumptions on the original continuous-parameter data. In the specific case of deinterlacing, and in the worst case, one can imagine a small feature one pixel high, moving at certain critical velocities, such that it is either always or never present on the interlaced grid. In real life, the velocity would not exactly match a critical velocity and so the feature would appear and disappear at a nonzero temporal frequency. If this feature is the edge of a rectangle or part of a straight line, the flickering can be noticed by the eye, but may be very hard to correct. Therefore, an issue in MC deinterlacers is the consistency of the estimate. A recursive block estimate showing a high degree of visual consistency is given in de Haan et al. [19].

11.4 BAYESIAN METHOD FOR ESTIMATING MOTION

In Chapter 8 we introduced Bayesian methods for image estimation and restoration, which use a Gibbs-Markov signal model together with some, most often iterative, solution method, such as simulated annealing (SA). Other methods used include deterministic iterative methods such as *iterated conditional mode* (ICM) and *mean field annealing* (MFA). ICM is a method that sweeps through the sites (pixels) and maximizes the conditional probability of each pixel in sequence. It is fast but tends to get stuck at local optima of the joint conditional probability, rather than find the global maxima. MFA is an annealing technique that assumes that the effect of a neighboring clique's potential function can be well modeled with its mean value. While this changes the detailed global energy function, the iteration proceeds much more quickly and reportedly provides very nice results, generally classified as being somewhere between ICM and SA. A general review of these approaches is contained in the review article by Stiller and Konrad [20].

To extend the Gibbs-Markov model, we need to model the displacement vector or motion field \mathbf{d} , which can be done as

$$f_{\mathbf{d}}(\mathbf{D}) = K \exp[-U_{\mathbf{d}}(\mathbf{D})],$$

where the matrix \mathbf{D} contains the values of the vector field \mathbf{d} on the image region or frame. The energy function $U_{\mathbf{d}}(\mathbf{D})$ is the sum of potential function over all the pixels (sites) \mathbf{n} and their corresponding cliques,

$$U_{\mathbf{d}}(\mathbf{D}) \triangleq \sum_{\mathbf{n}} \sum_{c_{\mathbf{n}} \in \mathcal{C}} V_{c_{\mathbf{n}}}(\mathbf{D}),$$

where $C_{\mathbf{n}}$ denotes the clique system for the displacement field \mathbf{d} , over frame \mathbf{n} .

A common setup calls for the estimation of the displacement between two frames, $\mathbf{X}_{\mathbf{n}}$ and $\mathbf{X}_{\mathbf{n}-1}$, and using the MAP approach, we seek the estimate

$$\hat{\mathbf{D}} = \arg \max_{\mathbf{D}} f(\mathbf{D} | \mathbf{X}_{\mathbf{n}}, \mathbf{X}_{\mathbf{n}-1}). \quad (11.4-1)$$

This simple model can be combined with a line field on an interpixel grid, as in Chapter 8, to allow for smooth estimates of displacement that respect the sharp

boundaries of apparent motion that occur at object boundaries. The assumption is that moving objects generally have the same velocity, while different objects, and the background, are free to move at much different velocities. An early contribution to such estimates is the work of Konrad and Dubois [21].

Now as was the case in Chapter 8, there are so many variables in (11.4–1) that simultaneous joint maximization is out of the question. On the other hand, iterative schemes that maximize the objective function one site at a time are practical. The nice thing about Gibbs models is that it is easy to find the local or marginal model. One simply gathers all the potential functions that involve that site. All the other terms go into the normalizing constant. Thus, in the ICM approach, we seek the peak of the resulting conditional pdf. In the SA technique, we take a sample from this pdf, proceed through all the sites, and then reduce the temperature a small amount. In MFA, we find the conditional mean of the conditional pdf, and iterate through all the sites in this manner.

Example 11.4–1: Motion Estimation for MC Prediction

This example comes from the review article [20] and shows both the predicted frame and the prediction error frame for three types of motion estimate: block-based, pixel-based, and region-based, the latter being based on segmenting the moving areas in the scene in some way (e.g., using line fields). The data come from the monochrome videophone test clip *carphone* in QCIF⁷ resolution. Figure 11.4–1 shows the result of block-based motion, using fairly large 16×16 blocks, with the block structure being clearly evident, especially around the mouth. Its prediction MSE is reported as 31.8 dB. The predicted frame in Figure 11.4–2, resulting from a dense motion estimate, is much better visually, with the much better prediction MSE of 35.9 dB. The region-based estimate, shown



FIGURE 11.4–1

Predicted frame of the carphone clip via block-based motion estimate. (©1999 IEEE)

⁷See appendix on video formats at the end of this chapter.

**FIGURE 11.4-2**

Predicted frame via dense motion estimate. (© 1999 IEEE)

**FIGURE 11.4-3**

Predicted frame via region-based estimate. (© 1999 IEEE)

**FIGURE 11.4-4**

Prediction error frame for block-based motion. (© 1999 IEEE)

**FIGURE 11.4-5**

Prediction error frame for dense motion. (© 1999 IEEE)

**FIGURE 11.4-6**

Prediction error frame for region-based motion. (© 1999 IEEE)

in Figure 11.4-3, is almost as good as the dense one but has many fewer motion vectors. Its prediction MSE is reported at 35.4 dB.

The corresponding prediction error frames are shown in Figures 11.4-4, 11.4-5, and 11.4-6. Regarding the prediction error frames for the dense and region-based motion, we can see some blurring of object edges caused by the dense motion estimate, yet the object boundaries are respected better in the region-based estimate. Details on this example are presented in [22].

Joint Motion Estimation and Segmentation

An alternative to motion estimation with a line field to prevent oversmoothing at object edges is to jointly estimate an object segmentation along with the motion [23].

The object boundary then serves to provide a linear feature over which motion should not be smoothed.

Example 11.4–2: A Joint Segment and Displacement Potential

Stiller and Konrad [20] provide the following example of a joint segmentation \mathbf{S} and displacement \mathbf{D} potential function. Here, the segment label $s(\mathbf{n})$ can take on a fixed number of label values $1, \dots, L$ corresponding to an assumed number of objects in the video frames. The potential for pairwise cliques was suggested as

$$V_{\mathbf{n}_1, \mathbf{n}_2}(\mathbf{d}(\mathbf{n}_1), \mathbf{d}(\mathbf{n}_2), s(\mathbf{n}_1), s(\mathbf{n}_2)) = \lambda_d \|\mathbf{d}(\mathbf{n}_1) - \mathbf{d}(\mathbf{n}_2)\|^2 \delta(s(\mathbf{n}_1) - s(\mathbf{n}_2)) \\ + \lambda_l (1 - \delta(s(\mathbf{n}_1) - s(\mathbf{n}_2))). \quad (11.4-2)$$

Here, λ_d and λ_l are weights and δ is the discrete-time impulse function. We see that if the labels are the same, the first term penalizes the potential if the motion vectors are different at the two neighbor pixels (sites) \mathbf{n}_1 and \mathbf{n}_2 . The second term in the potential penalizes the case where the labels are different at these two neighbor sites. The overall potential function then provides a compromise between these two effects.

One fairly complete formulation of the joint motion-segmentation problem is in the thesis of Han [24], where he applied the resulting estimates to video frame-rate increase and low bitrate video coding. His energy function followed the approach of Stiller [23] and was given as

$$(\hat{\mathbf{D}}_n, \hat{\mathbf{S}}_n) = \arg \max_{\mathbf{D}_n, \mathbf{S}_n} f(\mathbf{D}_n, \mathbf{S}_n | \mathbf{X}_n, \mathbf{X}_{n-1}) \\ = \arg \max_{\mathbf{D}_n, \mathbf{S}_n} f(\mathbf{X}_{n-1} | \mathbf{D}_n, \mathbf{S}_n, \mathbf{X}_n) f(\mathbf{D}_n | \mathbf{S}_n, \mathbf{X}_n) f(\mathbf{S}_n | \mathbf{X}_n), \quad (11.4-3)$$

where the three factors are the pdf's or pmf's of the *video likelihood model*, *motion field \mathbf{D}_n prior model*, and *segmentation field \mathbf{S}_n prior model*, respectively.

The likelihood model was Gibbsian with energy function,

$$U(\mathbf{X}_{n-1} | \mathbf{D}_n, \mathbf{X}_n) = \frac{1}{2\sigma^2} \sum_{\mathbf{n}} (x(\mathbf{n}, n) - x(\mathbf{n} - \mathbf{d}(\mathbf{n}, n), n-1))^2,$$

which is simplified to not depend on the segmentation.

The motion field prior model is also Gibbsian and given in terms of energy function,

$$U_d(\mathbf{D}_n | \mathbf{S}_n) = \lambda_1 \sum_{\mathbf{n}} \sum_{\mathbf{m} \in \mathcal{N}_n} \|\mathbf{d}(\mathbf{n}, n) - \mathbf{d}(\mathbf{m}, n)\|^2 \delta(s(\mathbf{n}, n) - s(\mathbf{m}, n)) \\ + \lambda_2 \sum_{\mathbf{n}} \|\mathbf{d}(\mathbf{n}, n) - \mathbf{d}(\mathbf{n} - \mathbf{d}(\mathbf{n}, n), n-1)\|^2 \\ - \lambda_3 \sum_{\mathbf{n}} \delta(s(\mathbf{n}, n) - s(\mathbf{n} - \mathbf{d}(\mathbf{n}, n), n-1)).$$

Here, he made the assumption that given the segmentation of the current frame, the motion field does not depend on the frame data. The first term in U_d sums over the local neighborhoods of each pixel and raises the motion field prior energy function if, and only if, the labels agree and the l^2 norm of the motions are different. The second term increases this energy if the motion vector has changed along the motion path to the previous frame. The third term *lowers* the energy function if the labels agree along the motion paths. Of course, careful selection of the lambda parameters is necessary to achieve the best trade-off among these various factors, which try to enforce spatial smoothness of motion along with temporal smoothness of motion and segmentation along the motion path.

The segmentation field prior model is given in terms of the energy function

$$U_s(S_n|X_n) = \sum_n \sum_{m \in N_n} V_m(s(n, n), s(m, n)|X_n),$$

where

$$V_m(s(n), s(m)|X_n) = \begin{cases} -\gamma, & s(n) = s(m) \text{ and } l(n) = l(m), \\ 0, & s(n) = s(m) \text{ and } l(n) \neq l(m), \\ +\gamma, & s(n) \neq s(m) \text{ and } l(n) = l(m), \\ 0, & s(n) \neq s(m) \text{ and } l(n) \neq l(m), \end{cases}$$

where $l(n)$ is a (deterministic) label field determined from X_n alone via a standard region-growing algorithm [24]. The overall intent is to encourage smooth

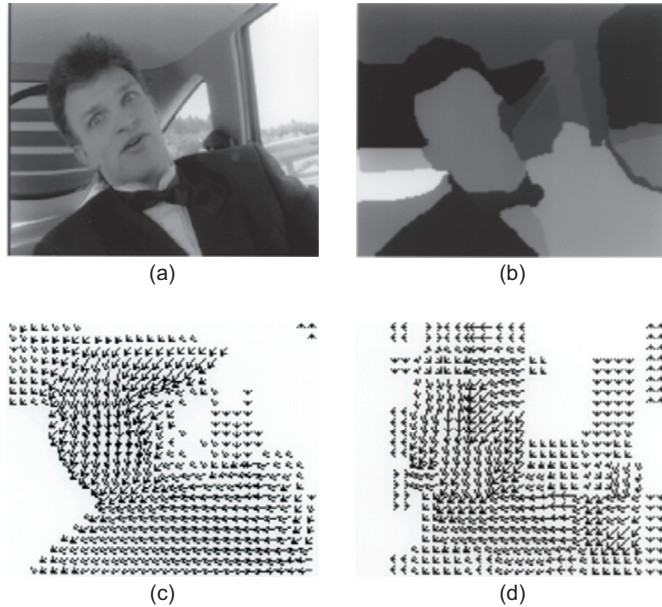


FIGURE 11.4-7

(a) Original, (b) joint segmentation, (c) block-based motion, (d) joint motion.

segmentation labels that agree with the 2-D label field determined just by color (gray level), penalize changes in segmentation field that do not agree with changes in color, and treat neutrally the remaining discrepancies. This is because the motion segmentation can change when an object starts or stops moving, and usually it should correspond to a color segmentation.

It is common to try to iteratively maximize (11.4-3) via an alternating method, where one sweep of all the sites tries to improve the criterion by updating $\hat{\mathbf{D}}_n$ given the current $\hat{\mathbf{S}}_n$, while the next or alternate sweep updates $\hat{\mathbf{S}}_n$ given the current $\hat{\mathbf{D}}_n$. The procedure starts with a segmentation determined by region growing on the first color frame. Figure 11.4-7 shows results of the alternating motion-segmentation optimization from [24]. We notice that the determined motion in Figure 11.4-7(d) is very smooth within the objects found in Figure 11.4-7(b), unlike the generally more noisy motion vectors in Figure 11.4-7(c) determined by block matching.

In the next chapter, we will show how joint motion and segmentation estimation can be used in an object-based low-bitrate video coder. This application can use the preceding formulation that assumes clean data are available.

11.5 RESTORATION OF DEGRADED VIDEO AND FILM

Another class of applications uses Bayesian methods to restore degraded image sequence data such as for restoring old movie film and video tape. Various kinds of problems can be present in addition to noise and blurring, including *blotches* or areas in each frame of missing data due to dirt, scratches, chemical deterioration, video tape dropout, etc. The observation model is given in terms of a binary masking function $b(\mathbf{n}, n)$ that indicates the presence of a blotch ($b = 1$) and a *replacement noise* model $v(\mathbf{n}, n)$ that models the statistics of the blotch, as well as the additive random noise $w(\mathbf{n}, n)$. The noisy and distorted observations $y(\mathbf{n}, n)$ are then written as

$$y(\mathbf{n}, n) = (1 - b(\mathbf{n}, n))x(\mathbf{n}, n) + b(\mathbf{n}, n)v(\mathbf{n}, n) + w(\mathbf{n}, n), \quad (11.5-1)$$

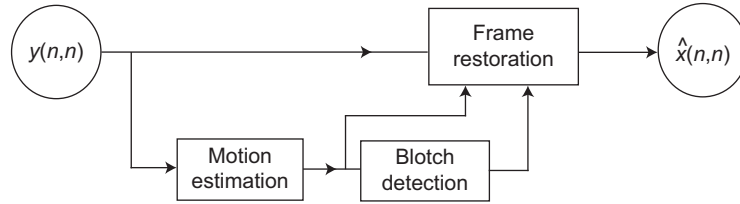
where $\mathbf{n} = (n_1, n_2)$. A simple approach to detect blotches first performs a robust and dense motion estimation, and then applies a spike detector to the energy in the motion-compensated residual. A spike-detector index (SDI) can then be given as

$$\text{SDI}(\mathbf{n}, n) = \min\{(y(\mathbf{n}, n) - y(\mathbf{n} - \mathbf{d}[\mathbf{n}; n, n - 1], n - 1))^2, \\ (y(\mathbf{n}, n) - y(\mathbf{n} - \mathbf{d}[\mathbf{n}; n, n + 1], n + 1))^2\},$$

where $\mathbf{d}[\mathbf{n}; n, n - 1]$ and $\mathbf{d}[\mathbf{n}; n, n + 1]$ represent the forward and backward motion-estimated displacements, respectively. This index is then compared to a threshold to detect a blotch as

$$b(\mathbf{n}, n) = \begin{cases} 0, & \text{SDI}(\mathbf{n}, n) \leq T, \\ 1, & \text{otherwise,} \end{cases}$$

where $T(> 0)$ is some set threshold. More on this topic appears in Chapter 4 of *The Essential Guide to Video* [15]. The detected blotch location together with the motion

**FIGURE 11.5–1**

An illustration of video restoration in the presence of blotch-type artifacts.

information is fed to an estimation stage where both are jointly applied in restoring the current frame to obtain the estimate $\hat{x}(n,n)$, as shown in Figure 11.5–1.

While this simple approach often works, it will face problems when motion fails, in part because the blotch is not explicitly considered when the motion is being estimated. Also, there is still the problem of estimating the clean image pixel $x(n,n)$, and explicit consideration of occlusions is omitted.

A Bayesian Approach

Kokaram has presented a comprehensive Bayesian approach to this joint detection/estimation problem [25] based on earlier work [26]. His model also uses (11.5–1) and includes motion and occlusion models used for predicting $x(n,n)$ from the nearest neighbor frames $y(n,n-1)$ and $y(n,n+1)$. Two other binary functions $o_b(n,n)$ and $o_f(n,n)$ model occlusion in backward and forward directions, respectively, with $o_i = 1$ indicating occlusion. A pixel state $s(n,n)$ is then given by $s \triangleq [b, o_b, o_f]$. The variables to be estimated at each pixel are thus

$$\theta(n,n) \triangleq [x(n,n), s(n,n), v(n,n)].$$

It has been observed that blotch corruption does not usually occur at the same location in consecutive frames; thus it is assumed that no blotch is present in frames $n-1$ or $n+1$ at the same location. The Bayesian *a posteriori* probability then becomes equivalent to the product of a likelihood term and a prior term, given as

$$\begin{aligned} p[\theta(n,n)|x(n,n-1), y(n,n), x(n,n+1)] &\propto p[y(n,n)|\theta(n,n), x(n,n-1), x(n,n+1)] \\ &\quad \times p[\theta(n,n)|x(n,n-1), x(n,n+1)]. \end{aligned} \quad (11.5-2)$$

Details on factoring and expanding this equation in terms of corruption likelihood and specific priors are given in [25], where an ICM rather than MAP solution method is used to limit computation. The motion is estimated initially using block matching, and initial blotch detection is made by a deterministic scheme. An iterative solution scheme then alternates between (1) an updated estimate of the current state $s(n,n)$ and image frame $x(n,n)$, given the current estimate of motion (displacement) $d(n,n)$ and observation noise variance $\sigma_w^2(n)$, and (2) a blockwise scan of the current

image frame n that updates the estimate of the displacement $\mathbf{d}(n, n)$, given the current estimate for state s and image x . Good experimental performance for this joint detection/estimation approach is reported.

11.6 SUPER-RESOLUTION OF VIDEO

Restoration from a small number of images by super-resolution (SR) methods was presented in Chapter 8, but the goal was to produce just one high-resolution (HR) image from the low-resolution (LR) input frames. Patti et al. [27] presented a quite general SR algorithm for conversion of SD to HD video based on a projections-onto-convex-sets formulation (see Section 5.1, Chapter 5) [28]. A dynamic or recursive formulation of SR for video was presented in Farsiu et al. [29] based on an earlier formulation of Elad and Feuer [30].

One way to handle the video case for SR would be to repeat the image SR solution for each frame in the video, but clearly this is not efficient or even practicable because of the implied growing memory of such an estimate. So, a recursive solution was sought. In [29], taking the monochrome case, they first set up the following dynamic equations:

$$\mathbf{X}_n = \mathcal{F}_{n|n-1}\mathbf{X}_{n-1} + \mathbf{U}_n, \quad (11.6-1)$$

$$\mathbf{Y}_n = \mathcal{D}\mathcal{H}\mathbf{X}_n + \mathbf{W}_n, \quad (11.6-2)$$

for the HR dynamic system (11.6-1) and LR observations (11.6-2), where $\mathcal{F}_{n|n-1}$ represents motion between frames (assumed known), \mathcal{D} is a decimation operator, \mathcal{H} is a blur operator, and \mathbf{U}_n and \mathbf{W}_n are independent model and observation noise frames, respectively. We again use the 2-D vector/4-D matrix notation introduced in problem 20 of Chapter 4. Note that this spatiotemporal model is in the *temporally causal sense* (see Section 10.4 in Chapter 10) since the prediction step in (11.6-1) only involves the prior frame, frame \mathbf{X}_{n-1} . A temporally causal Kalman filter could be defined here with a global state equal in size to a full image frame, analogously to the totally ordered temporally causal Kalman filter developed in Section 10.4.

Before presenting their estimator, however, they perform a separation of the SR problem into two parts: a data-fusing part and a restoration part, similar to the case in Section 8.8 of Chapter 8. They point out that, upon definition of the blurred video $\mathbf{Z}_n \triangleq \mathcal{H}\mathbf{X}_n$, the preceding equations can be written in terms of the 2-D state vector \mathbf{Z}_n on the HR grid, as

$$\mathbf{Z}_n = \mathcal{F}_{n|n-1}\mathbf{Z}_{n-1} + \mathbf{V}_n, \quad (11.6-3)$$

$$\mathbf{Y}_n = \mathcal{D}\mathbf{Z}_n + \mathbf{W}_n, \quad (11.6-4)$$

where $\mathbf{V}_n = \mathcal{H}\mathbf{U}_n$, thus permitting the problem to be decomposed into a data-fusing step giving \mathbf{Z}_n and then the conventional filtering equation

$$\mathbf{Z}_n = \mathcal{H}\mathbf{X}_n.$$

From this decomposition, they decide to first solve the estimation problem (11.6-3 and 11.6-4) to get the estimate $\hat{\mathbf{Z}}_n$, and then to solve a restoration problem to find $\hat{\mathbf{X}}_n$, the desired estimate of the HR video \mathbf{X}_n .

The estimator for the first problem (11.6-3 and 11.6-4) is then chosen as a single update version of the temporally causal Kalman filter and is of the form

$$\hat{\mathbf{Z}}_n = \mathcal{F}_{n|n-1} \hat{\mathbf{Z}}_{n-1} + \mathcal{K}_n (\mathbf{Y}_n - \mathcal{D} \mathcal{F}_{n|n-1} \hat{\mathbf{Z}}_{n-1}),$$

where the update or gain operator \mathcal{K}_n is diagonal and given in [29] in terms of a recursion based on an assumed diagonal form for $\text{cov}(\mathbf{V}_n)$. We note that the concatenated operator $\mathcal{D} \mathcal{F}_{n|n-1}$ is just a decimate and shift operator and hence relatively easy to implement. Since the Kalman gain operator (4-D matrix) is diagonal, there is only one update per pixel in their procedure. But notice the key role of the motion compensation or shifting operator $\mathcal{F}_{n|n-1}$ in creating this estimate. It effectively decides what LR pixels will be included according to whether the composite shifts move into the field of view of \mathbf{Z}_n .

The second step is to estimate the HR video \mathbf{X}_n from the estimate $\hat{\mathbf{Z}}_n$, and this is done using a robust deblurring method in [29] as

$$\min_{\mathbf{X}_n} \left\{ \|\mathcal{H} \mathbf{X}_n - \hat{\mathbf{Z}}_n\|_2^2 + \lambda \Gamma(\mathbf{X}_n) \right\}, \quad (11.6-5)$$

where the constraint term $\Gamma(\mathbf{X}_n)$ is given in terms of a locally weighted L_1 norm of a spatial regularization term

$$\Gamma(\mathbf{X}_n) = \sum_{\max(|m_1|, |m_2|) \leq p} \alpha^{|m_1| + |m_2|} \|\mathbf{X}_n - \mathcal{S}_1^{m_1} \mathcal{S}_2^{m_2} \mathbf{X}_n\|_1, \quad (11.6-6)$$

and where \mathcal{S}_1 and \mathcal{S}_2 are horizontal and vertical shift operators, respectively. The parameters $\lambda > 0$, $0 < \alpha < 1$ and small positive number p are experimentally chosen to optimize a steepest descent solution for (11.6-5).

For this algorithm to be of practical importance, there must be available an efficient subpixel accurate motion estimation procedure to supply the assumed known displacement \mathcal{F} operator on the LR frames. Since the algorithm has no knowledge of (does not account for) motion errors, the motion estimate will have to be highly accurate as well. Since the LR frames are significantly aliased, in order to be able to expect significant SR performance improvements, the required motion estimation may be a challenge. As mentioned in Section 8.8 of Chapter 8, motion estimation methods specifically adapted to the presence of aliasing may be needed [31].

Demosaicing

One place where there can be significant alias energy is in a digital image captured with a Bayer color filter array (CFA) (see Section 6.6, Chapter 6). The processing needed to restore the missing pixels is called demosaicing, and there have been many attempts to demosaic still images [32]. Such processing has not always succeeded due to the ill-conditioned and under-determined nature of this problem. However, processing a sequence of such images, with known displacements, while keeping the

basic field of view of the camera intact, can much improve the situation by providing the missing samples as well as reducing the observation noise, by averaging over the frames along the motion trajectories.

The basic fusion step of estimating the frames \mathbf{Z}_n can remain as before, with a separate estimate for each color component R , G , and B . Then in the restoration step of estimating \mathbf{X}_n from $\hat{\mathbf{Z}}_n$, a weighted L^2 norm can be formulated as

$$\left\| \mathcal{A}_R \left(\mathcal{H} \mathbf{X}_n^{(R)} - \hat{\mathbf{Z}}_n^{(R)} \right) \right\|_2^2 + \left\| \mathcal{A}_G \left(\mathcal{H} \mathbf{X}_n^{(G)} - \hat{\mathbf{Z}}_n^{(G)} \right) \right\|_2^2 + \left\| \mathcal{A}_B \left(\mathcal{H} \mathbf{X}_n^{(B)} - \hat{\mathbf{Z}}_n^{(B)} \right) \right\|_2^2,$$

where the weights are diagonal operators (diagonal 4-D matrices) that have zero weights for those pixels with no samples [29]. This error term was regularized with the addition of three penalty terms: an L^1 penalty term on the luminance, following (11.6–6), an L^2 penalty term on the Laplacian of two chrominance terms, and an innovative orientation penalty term [29] that tends to orient edges similarly across the three color channels. Steepest descent is recommended for the solution.

Example 11.6–1: SR of Mosaiced Video

For a color camera that uses a Bayer array, there is a loss or resolution due to the CFA. The resulting subsampling operator \mathcal{D} can be illustrated as in Figure 11.6–1 from [29], which shows seven LR mosaiced color image frames and their hypothetical position on a super-resolved HR image.

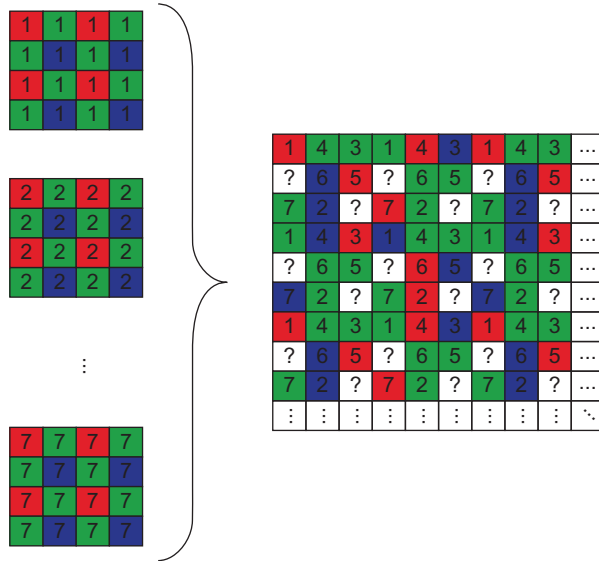
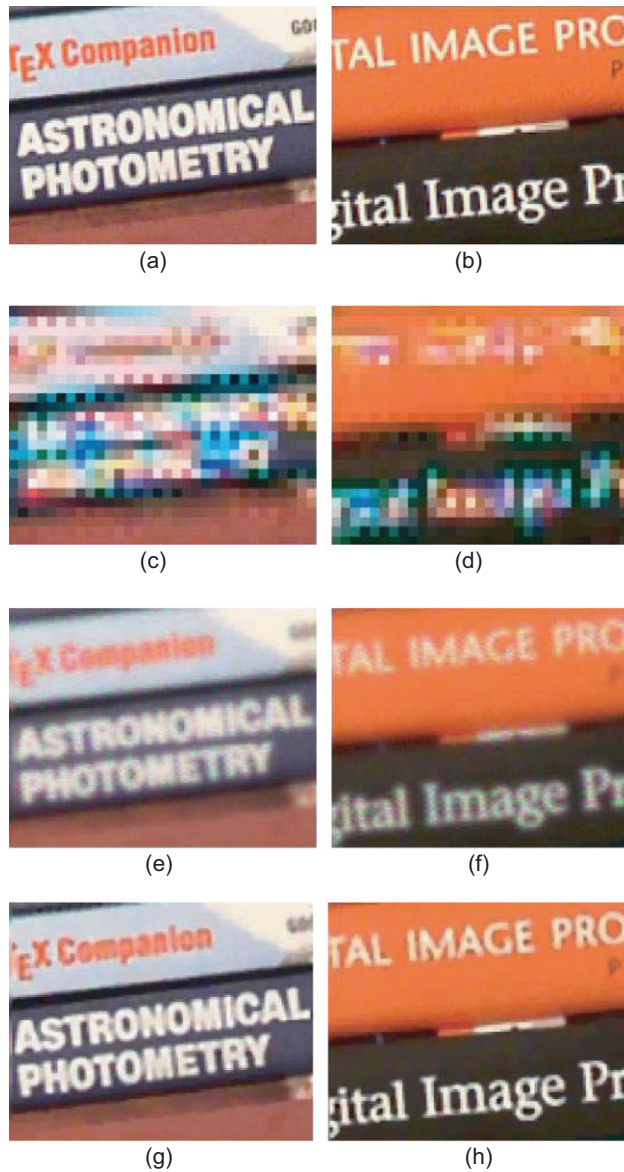


FIGURE 11.6–1

An illustration of color filter subsampling effect. (Figure 11.1–3 from [29] with permission).

**FIGURE 11.6-2**

Some results of the Farsiu et al. study. (Figure 11.1-5 [29]).

A simulation study was conducted in [29] involving a 128×128 section of a large color image that was blurred, shifted one pixel at a time up, down, or right, and then subsampled 4×4 , effectively taking a random walk through the large original image. A

Bayer color filter was then applied to each LR frame, followed by the addition of white Gaussian noise to achieve an $\text{SNR} = 30$ dB. A total of 250 LR frames were created in this way. The images in Figure 11.6–2 show the results, with parts (a) and (b) being the original images of frames 50 and 250, respectively. Parts (c) and (d) are the corresponding LR images, (e) and (f) are the recursively fused HR frames, and (g) and (h) are the restored and demosaiced frames.

Note that the SR terminology can be applied to other video processing as well, e.g., 480i and 1080i deinterlacing. These operations, by their nature, are SR; that is, they come up with estimates for the missing samples by shifting nearby samples guided by accurate motion estimation/compensation. A possible new application of SR is to the subsampled HD video coming from current DSLR cameras. Possible application to motion-compensated filtering in the presence of subpixel accuracy should also be possible.

Finally, we point out that recently the nonlocal means approach (see Section 8.7, Chapter 8) has been extended to super-resolution [33] in a way that can alleviate the need for explicit motion estimation in SR processing. Using nonlocal means, it is possible to achieve super-resolution from a single image [34].

CONCLUSIONS

Multidimensional filtering plays an important role in video format conversion. Whenever an image or video is resized to fit a given display, a properly designed multidimensional filter will improve results over commonly employed decimation and 1-D filtering alone. Motion-compensated filtering turns out to be very effective on noisy or distorted video data, for the purpose of restoration. Motion-compensated multidimensional filtering also offers increased sharpness in video standards conversion. However, unless the MC estimates are very good, artifacts can result, and this is called *motion model failure*. This is where a simple translational or affine image model does not match the data that well. We then looked at Bayesian methods for joint motion estimation and segmentation. We overviewed the application area of restoring old film and video. Finally, we presented super-resolution of video, which makes use of accurate motion to increase the resolution of trackable moving objects in the input video.

PROBLEMS

1. How many pixels per second must be processed in standard definition D1 video?⁸ How much storage space in gigabytes (GB) for 10 minutes of D1?

⁸A description of the D1 format is provided in the appendix of this chapter.

2. Use MATLAB to design a spatial lowpass filter and then optimize it visually with the approach of [Example 11.1–3](#). Can you see the difference? What value of α did you determine? How does it relate to the transition bandwidth of the original lowpass filter?
3. Compare the computation (multiplies and adds) of full-search versus three-step search block matching, both using the MSE method. Assume an $M \times M$ block and $N \times N$ search window, and that block-matching estimates are calculated on an $M \times M$ decimated grid. Take the image size as CIF (i.e., 352×288).
4. In overlapping block motion compensation (OBMC), the weighting matrices of [10] are used to smooth the motion vectors over the fixed block boundaries. Comment on how this would be expected to affect the computational complexity of OBMC versus regular block matching.
5. In this problem you will show the connection between the spatiotemporal constraint equation (11.2–4) and direct minimization of the error,

$$E(d_x, d_y) \triangleq \sum_{R(x,y)} [f(x, y, t) - f(x - d_x, y - d_y, t - \Delta t)]^2.$$

First, calculate $\partial E / \partial d_x$ and $\partial E / \partial d_y$ and set them both to zero. Then argue that if at least one of the spatial gradients $\partial f / \partial x$ and $\partial f / \partial y$ are not zero, then for sufficiently small region $R(x, y)$ and time interval $(t, t + \Delta t)$, the spatiotemporal constraint equation must hold, with $d_x = v_x \Delta t$ and $d_y = v_y \Delta t$. (Hint: expand the function $f(x, y, t)$ in a Taylor series and keep only the first-order terms.)

6. Show in block-diagram form a system realizing the diamond filter deinterlacing method of [Example 11.3–3](#). Your diagram should explicitly include the up-samplers and the filter. For ATSC 1080i format, approximately how many multiplies per second will be required?
7. Extending [Example 11.3–5](#) on motion-compensated deinterlacing, we can look back over several field pairs (frames) and find multiple references for the missing pixel. Then the estimate for the missing pixel can be a weighted average of the references from the different frames. Suggest a good way of weighting these multiple references? You may care to look at [35] for some ideas.
8. Can mean field annealing (MFA) be used to estimate the segment labels in a joint motion-segmentation approach to Bayesian motion estimation? How would an iterated conditional mode (ICM) estimate be obtained?
9. Separate the three energy functions corresponding to the Gibbs pdf/pmf factors in (11.4–3) into two partially overlapping sums: one with all the $s(\mathbf{n}, n)$ terms and the other with all the $\mathbf{d}(\mathbf{n}, n)$ terms.
10. According to the constant intensity assumption for motion, we have

$$x(n_1, n_2, n) = x(n_1 - d_1, n_2 - d_2, n - 1),$$

where we assume that the displacement vector \mathbf{d} has integer elements d_1 and d_2 . Thus, by induction, we can write the video (image sequence) signal x as

$$\begin{aligned} x(n_1, n_2, n) &= x(n_1 - nd_1, n_2 - nd_2, 0) \\ &= g(n_1 - nd_1, n_2 - nd_2), \end{aligned}$$

for the 2-D function $g(n_1, n_2) \triangleq x(n_1, n_2, 0)$ and for a sequence of images starting with the first frame at $n = 0$.

- (a) Find and express the 3-D Fourier transform $X(\omega_1, \omega_2, \omega)$ in terms of the 2-D Fourier transform $G(\omega_1, \omega_2)$ and the integer displacements d_1 and d_2 . Assume there is no aliasing.
 - (b) Assuming given positive integer values of displacements d_1 and d_2 , what isotropic limit on the spatial bandwidth of g is necessary to avoid temporal frequency aliasing in x ? Use $|\omega|_{\max}$ to denote the spatial bandwidth of g , $0 < |\omega|_{\max} < \pi$.
11. Show that triangular mesh matching with affine models produces a continuous-velocity field across triangular patch edge boundaries, independent of the non-shared control point velocities. For example, in Figure 11.P-1 with the indicated control point velocities \mathbf{v}_i , both the upper and lower triangular patches share \mathbf{v}_1 and \mathbf{v}_4 but do not share \mathbf{v}_2 and \mathbf{v}_3 . Note the (x, y) origin at the upper left.

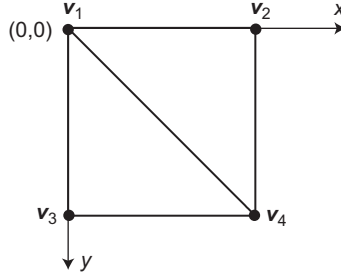


FIGURE 11.P-1

One cell of a triangular mesh with indicated velocities $\mathbf{v}_i, i = 1 - 4$, at the control points.

12. Obtain a copy of Kokaram's 2004 paper [25] and complete the expansion of the MAP expression (11.5-2).

APPENDIX: DIGITAL VIDEO FORMATS

This appendix presents some common video formats. We start with the common half-resolution formats SIF and CIF and then discuss the *standard-definition* (SD) format ITU 601. In passing, we consider a smaller quarter-size format called QCIF used

for low-bitrate video conferencing. We also list current *high-definition* (HD) formats specified by the ATSC and the digital cinema formats of DCI.

SIF

The *source interchange format* (SIF) is the format of MPEG1. It has 352×240 pixels at 30 fps, in a noninterlaced or progressive lattice. This is the U.S. version, more formally called SIF-525. The European version, SIF-625, is 352×288 . The MPEG committee has given recommendations for the conversion of ITU 601 to SIF and for the conversion of SIF to ITU 601 using certain suggested filters. These filters are commonly called *MPEG half-band filters*.

Decimation filter:

$$-29 \quad 0 \quad 88 \quad 138 \quad 88 \quad 0 \quad -29 // 256^9$$

Interpolation filter:

$$-12 \quad 0 \quad 140 \quad 256 \quad 140 \quad 0 \quad -12 // 256$$

CIF

The *common intermediate format* (CIF) has 352×288 -pixel, progressive image frames, with a sometimes variable frame rate. CIF gets its dimensions from the larger of these two (i.e., SIF-525 and SIF-625). Also common is *quarter CIF* (QCIF), the format of H.263, a video telephone 176×144 standard that runs at typically 5–15 fps. Finally, going in the direction of higher resolution, there is 4CIF with resolution 704×576 . Together, 4CIF, CIF, and QCIF make an embedded set of resolutions in spatial resolution.

ITU 601 Digital SDTV (a.k.a. SMPTE D1 and D5)

This is the component digital standard of the D1 (3/4 in.) digital SD tape recorders. The signal is sampled in 4:2:2 Y', C_R, C_B format, and interlaced 2:1. The field rate is 59.94 Hz (in the U.S.), with the frame rate 29.97 Hz (in the U.S.). The aspect ratio is 4:3, with the bit depth of 8 or 10 bits per pixel.

The intent was to digitize SD analog TV as seen in the United States and Europe. This standard is meant for viewing distance of 5–6 times the picture height. There are various members of the ITU 601 family, the most prominent being the so-called 4:2:2 *member*. The sampling frequency was standardized at 13.5 MHz for both 525 and 625 line systems (i.e., “4” means 13.5 MHz). This choice results in a static orthogonal sampling pattern for both 525/60 and 625/50 systems, since 13.5 and 6.75 MHz are integer multiples of 2.25 MHz, the least common multiple of the scan line frequencies of these systems. There are 720 active luma samples/line and 486 active lines (i.e., 720×486 pixels in the U.S. version at 60 fields/sec, and in Europe, 720×576 pixels at 50 fields/sec). The quantization is uniform PCM, again to either 8 or 10 bits.

⁹To get the actual coefficients, divide the indicated integer value by 256.

The standard provides filter specs for both the sampling of luma and for the subsampling of the chroma channels. The overall data rate was specified as 27 MHz.

From the digital point of view, there are 486 real or active lines, not the 525 lines in the U.S. SD system. The 525 line figure includes so-called “invisible lines,” which allowed for an analog CRT to “retrace” or return to both the beginning of a line (horizontal retrace interval) and return to the top left corner for display of the next field or frame (vertical retrace).

The Society of Motion Picture and Television Engineers (SMPTE) recommended the following color difference components or color space:

$$\begin{bmatrix} Y' \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.500 & -0.418 & -0.081 \\ -0.169 & -0.331 & 0.500 \end{bmatrix} \begin{bmatrix} R'_{709} \\ G'_{709} \\ B'_{709} \end{bmatrix},$$

where the R'_{709} , G'_{709} , and B'_{709} components are gamma corrected as in Section 6.5 of Chapter 6. In the common 4:2:2 mode, the C_R and C_B samples are co-sited with odd samples of Y' on each line (i.e., 1st, 3rd, 5th, etc.). There is a highest quality 4:4:4 mode where there is no color subsampling. The 4:4:4 mode is generally used in graphics and *matting* work, where an image is superposed on a different background.

What is 4:2:0?

The 4:2:0 specification means that the C_R and C_B chroma signals are subsampled by two vertically as well. This gives a combined chroma sample rate of 50% of the luma sample rate. Whenever there is subsampling of the color information, as in 4:2:2 and 4:2:0 rasters, there is a need to specify any offsets between the chroma and luma rasters. Converting from one representation to another is best done using multidimensional filters. This 4:2:0 color format is used in DV and HDV video tape formats, as well as DVD video.

ATSC Broadcast Formats

The digital formats in the United States were set up by the *Advanced Television System Committee* (ATSC) in the early 1990s. They mainly consist of two HD formats and two SD formats. The SD formats are 720×486 interlaced at 60 fields/sec and 720×486 progressive at 60 fps. The main HD formats are 1280×720 at 60 fps, often referred to as 720p, and 1920×1080 interlaced at 60 fields/sec, often referred to as 1080i. In recent years, a 1080p specification has also emerged also at 60 fps, with an uncompressed data rate of 3 Gbps, twice that of 1080i. Two bit depths are supported for the color components, 8 and 10 bits per sample. The ATSC keeps a Web site at <http://www.atsc.org> where further information on these formats can be downloaded. The SMPTE D5 HD tape provides a high-quality intraframe recording capability for these ATSC video standards.

DCI Formats

The DCI format (http://en.wikipedia.org/wiki/Digital_Cinema_Initiatives) was first issued in 2005 but has been updated since, with the latest version being 1.2

(http://www.dcmovies.com/DCIDigitalCinemaSystemSpecv1_2.pdf), which is the de facto standard for digital cinema in the United States. It describes the so-called digital cinema distribution master (DCDM) format, compression method, transport, security, projection systems, etc.

The image format is specified as either 2K (2048×1024) at 24 or 48 fps or 4K (4096×2160) at 24 fps. Within these formats, one can specify what pixels are to be displayed, the *active pixels*. For example, a movie with a 2.39 aspect ratio can fit in the 4K format as 4096×1716 or in the 2K format as 2048×858 . For this reason, the 4K and 2K distribution formats are sometimes referred to as 4K and 2K *containers*. The color space is specified as $X'Y'Z'$ with a bit depth of 12 bits and no color subsampling. Additionally, there is a hierarchical image structure that permits a 2K projection system to decode a 2K version from the 4K files. The resolution-embedded JPEG 2000 image coder has been adopted for intraframe compression of digital cinema. Special profiles of JPEG 2000, Part 1 have been developed for digital cinema.

REFERENCES

- [1] J. W. Woods and J. Kim, "Motion Compensated Spatiotemporal Kalman Filter," Chapter 12 in *Motion Analysis and Image Sequence Processing*, Eds. I. Sezan and R. L. Lagendijk, Kluwer, Boston, MA, 1993.
- [2] E. Dubois and W. Schreiber, "Improvements to NTSC by Multidimensional Filtering," vol. 97, *SMPTE J.*, pp. 446–463, June 1988.
- [3] H. Schröder and H. Blume, *One- and Multidimensional Signal Processing*, John Wiley and Sons, 2000.
- [4] H. Schröder, "On Vertical Filtering for Flicker Free Television Reproduction," *IEEE Trans. Circuits, Sys., and Signal Process.*, vol. 3, pp. 161–176, 1984.
- [5] J. R. Jain and A. K. Jain, "Displacement Estimation and Its Application in Interframe Image Coding," *IEEE Trans. Comm.*, vol. COM-29, pp. 1799–1808, December 1981.
- [6] H.-M. Hang and Y.-M. Chou, "Motion Estimation for Image Sequence Compression," Chapter 5 in *Handbook of Visual Comm.*, H.-M. Hang and J. W. Woods, Eds., Academic Press, New York, 1995.
- [7] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimating of Block Motion Vectors," *IEEE Trans. CSVT*, vol. 3, pp. 148–157, April 1993.
- [8] R. Thoma and M. Bierling, "Motion-Compensated Interpolation Considering Covered and Uncovered Background," *Signal Process.: Image Comm.*, Elsevier, vol. 1, no. 2, pp. 191–212, October 1989.
- [9] S. Nogaki and M. Ohta, "An Overlapped Block Motion Compensation for High Quality Motion Picture Coding," *Proc. IEEE Int. Sympos. Circuits Systems*, pp. 184–187, May 1992.
- [10] M. T. Orchard and G. J. Sullivan, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach," *IEEE Trans. Image Process.*, vol. 3, pp. 693–699, September 1994.
- [11] A. N. Netravalli and J. D. Robbins, "Motion Compensated Coding: Some New Results," *Bell Sys. Tech. J.*, vol. 59, no. 9, pp. 1735–1745, November 1980.

- [12] M. Bierling and R. Thoma, "Motion Compensating Field Interpolation Using a Hierarchically Structured Displacement Estimator," *Signal Processing*, vol. 11, pp. 387–404, December 1986.
- [13] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intell.*, vol. 17, pp. 185–203, August 1981.
- [14] H. Farid and E. P. Simoncelli, "Differentiation of Discrete Multidimensional Signals," *IEEE Trans. Image Process.*, vol. 13, pp. 496–508, April 2004.
- [15] Chapters in *The Essential Guide to Video*, A. Bovik, Ed., Elsevier/Academic Press, Burlington, MA, 2009.
- [16] R. D. Forni and D. S. Taubman, "On the Benefits of Leaf Merging in Quad-tree Motion Models," *Proc. IEEE ICIP*, vol. 2, pp. 858–861, September 2005.
- [17] M. K. Ozkan, I. Sezan, A. T. Erdam, and A. M. Tekalp, "Motion Compensated Multi-frame Wiener Restoration of Blurred and Noisy Image Sequences," *Proc. IEEE ICASSP*, vol. 3, pp. 293–296, San Francisco, CA, March 1992.
- [18] J. W. Woods and S.-C. Han, "Hierarchical Motion Compensated De-interlacing," *Proc. SPIE Visual Communications and Image Processing (VCIP) VI*, vol. 1605, no. 805, Boston, November 1991.
- [19] G. de Haan, P. W. A. C. Biezen, H. Huijgen, and O. A. Ojo, "True-Motion Estimation with 3-D Recursive Search Block Matching," *IEEE Trans. CSVT*, vol. 3, pp. 368–379, October 1993.
- [20] C. Stiller and J. Konrad, "Estimating Motion in Image Sequences," *IEEE Signal Process. Magazine*, vol. 6, pp. 70–91, July 1999.
- [21] J. Konrad and E. Dubois, "Estimation of Image Motion Fields: Bayesian Formulation and Stochastic Solution," *Proc. IEEE ICASSP*, vol. 2, pp. 1072–1075, April 1988.
- [22] V.-B. Dang, A. R. Mansouri, and J. Konrad, "Motion Estimation for Regional-Based Video Coding," *Proc. IEEE ICIP*, vol. 2, pp. 198–192, Washington, DC, October 1995.
- [23] C. Stiller, "Object Based Estimation of Dense Motion Fields," *IEEE Trans. Image Process.*, vol. 6, pp. 234–250, February 1997.
- [24] S.-C. Han, *Object-Based Representation and Compression of Image Sequences*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1997.
- [25] A. C. Kokaram, "On Missing Data Treatment for Degraded Video and Film Archives: a Survey and a New Bayesian Approach," *IEEE Trans. Image Process.*, vol. 13, pp. 397–415, March 2004.
- [26] A. C. Kokaram and S. J. Godsill, "MCMC for Joint Noise Reduction and Missing Data Treatment in Degraded Video," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 189–205, February 2002.
- [27] A. J. Patti, M. I. Sezan, and A. M. Tekalp, "Superresolution Video Reconstruction with Arbitrary Lattices and Nonzero Aperture Time," *IEEE Trans. Image Process.*, vol. 6, no. 8, pp. 1064–1076, August 1997.
- [28] H. Stark and P. Oskoui, "High Resolution Image Recovery from Image Plane Arrays Using Convex Projections," *J. Optical Soc. Amer. A*, vol. 6, pp. 1715–1726, 1989.
- [29] S. Farsiu, M. Elad, and P. Milanfar, "Video-to-Video Dynamic Super-Resolution for Grayscale and Color Sequences," *EURASIP J. on Advances in Signal Processing*, Hindawi, vol. 2006, article ID 61859, pp. 1–15, 2006.
- [30] M. Elad and A. Feuer, "Super-Resolution Restoration of an Image Sequence: Adaptive Filtering Approach," *IEEE Trans. Image Process.*, vol. 8, no. 3, pp. 387–395, March 1999.

- [31] P. Vandewalle, S. Susstrunk, and M. Vetterli, "A Frequency Domain Approach to Registration of Aliased Images with Application to Super Resolution," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 1–14.
- [32] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Color Plane Interpolation Using Alternating Projections," *IEEE Trans. Image Process.*, vol. 11, no. 9, pp. 997–1013, September 2002.
- [33] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the Nonlocal-Means to Super-Resolution Reconstruction," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 36–51, January 2009.
- [34] D. Glasner, S. Bagon, and M. Irani, "Super-Resolution from a Single Image," *Proc. Int. Conf. Comput. Vision (ICCV)*, pp. 349–356, Kyoto, Japan, September 2009.
- [35] T. Wiegand, X. Zhang, and B. Girod, "Long-term Memory Motion-Compensated Prediction," *IEEE Trans. CSVT*, vol. 9, pp. 70–84, February 1999.

Digital Video Compression

12

Video coders compress a sequence of images, so this chapter is closely related to Chapter 9 on image compression. Many of the techniques introduced there will be used and expanded upon in this chapter. At the highest level of abstraction, video coders comprise two classes: *interframe* and *intraframe*, based on whether they use statistical dependency between frames or are restricted to using dependencies only within a frame. The most common intraframe video coders use the DCT and are very close to JPEG; for video, they are called M-JPEG, wherein the “M” can be thought of as standing for “motion,” as in “motion picture.” Also, common intraframe video compression is the DV coder in standard-definition video camcorders. By contrast, interframe coders exploit dependencies across frame boundaries to gain increased efficiency. The most efficient of these coders make use of the apparent motion between video frames to achieve their significantly larger compression ratio. An interframe coder will generally code the first frame using intraframe coding but will then use interframe coding on the other frames. Common interframe video coders are HDV and AVCHD consumer camcorders.

MPEG coders restrict their interframe coding to a *group of pictures* of relatively small size, say 12–60 frames, to prevent error propagation. These MPEG coders use a transform compression method for both the frames and prediction residual data, thus exemplifying *hybrid coding*, since the coder is a hybrid of the block-based transform spatial coder of Chapter 9 and a predictive or DPCM temporal coder. The current standards-based coder H.264/AVC uses a mixture of transform and spatial prediction on the prediction residuals. When transforms are used in both spatial and time domains, the coder is commonly called a 3-D *transform coder*.

All video coders share the need for a source buffer to smooth the output of the variable-length coder for each frame. The overall video coder can be *constant bitrate* (CBR) or *variable bitrate* (VBR) depending on whether the buffer output bitrate is constant. Often, intraframe video coders are CBR, in that they assign or use a fixed number of bits for each frame. In the case of interframe video coders, the bitrate is more highly variable. For example, an action sequence may need a much higher bitrate to achieve a good quality than would a so-called “talking head,” but this is only apparent from the interframe viewpoint.

Table 12.0–1 Types of Video with Uncompressed and Compressed Bitrates				
Video	Pixel Size	Frame Rate	Rate (uncomp.)	Rate (comp.)
Teleconference (QCIF)	176 × 144	5–10 fps	3–6 Mbps	32–64 Kbps
Multimedia (SIF)	352 × 288	30	61 Mbps	200–300 Kbps
Standard definition (SD)	720 × 480	30	243 Mbps	4–7 Mbps
High definition (HD)	1920 × 1080	24, 30	1.0 Gbps	20–35 Mbps
Digital cinema (DC)	4096 × 2160	24	7.6 Gbps	100–200 Mbps

Table 12.0–1 shows the various types of digital video, along with frame size in pixels and frame rate in frames per second (fps). Uncompressed bitrate assumes an 8-bit pixel depth, except for 12-bit digital cinema, and includes a factor of three for RGB color. The last column gives an estimate of compressed bitrates using technology, such as H.264/AVC and MPEG 2. We see fairly impressive compression ratios comparing the last two columns of the table, in the vicinity of 100 in several cases. While the given “pixel size” may not relate directly to a chosen display size, it does give an indication of recommended display size for general visual content, with standard definition (SD) and above generally displayed at full screen height, multimedia at half screen height, and teleconference displayed at one-quarter screen height, on a normal display terminal. In terms of distance from a viewing screen, it is conservatively recommended to view SD at 6 times the picture height, and HD at three times the picture height, and DC at 1.5 times the picture height (see Chapter 6).

12.1 INTRAFRAME CODING

In this section we look at three popular intraframe coding methods for video. They are block DCT, motion M-JPEG, and SWT. The new aspect over the image coding problem is the need for rate control, which arises because we may want to have a variable bit assignment across the frames to get more uniform quality. In fact, if we would like to have constant quality across the frames that make up our video, then the bit assignment must adapt to frame complexity, resulting in a VBR output of the coder.

Figure 12.1–1 shows the system diagram of a transform-based intraframe video coder. We see the familiar transform, quantize, and VLC structure. What is new is the buffer on the output of the VLC and the feedback of a rate control from the buffer. Here, we have shown the feedback as controlling the quantizer. If we need CBR video, then the buffer output bitrate must be constant, so its input bitrate must be controlled so as to avoid overflow or underflow, the latter corresponding to the case where this output buffer is empty and therefore unable to supply the required CBR. The common way of controlling the bitrate is to monitor buffer fullness and then feed back this information to the quantizer. Usually the step size of the quantizer is adjusted to keep the buffer fullness around its midpoint, or half full.

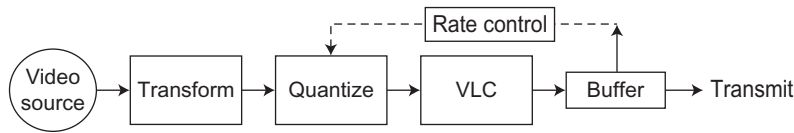
**FIGURE 12.1-1**

Illustration of intraframe video coding with rate control.

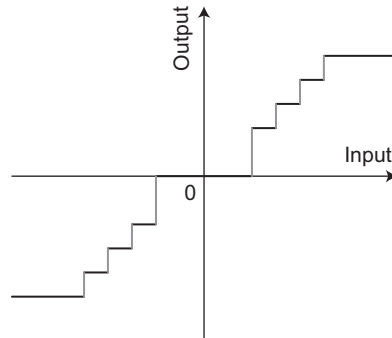
**FIGURE 12.1-2**

Illustration of uniform threshold quantizer (UTQ), named for its deadzone at the origin.

As mentioned in Section 9.3, a uniform quantizer has a constant step size, so that if the number of output levels is an odd number and the input domain is symmetric around zero, then zero will be an output value. If we enlarge the step size around zero, say by a factor of two, then the quantizer is said to be a *uniform threshold quantizer* (UTQ) (Figure 12.1-2). The bin that gets mapped into zero is called the *dead zone* and can be very helpful to reduce noise and “insignificant” details. Another nice property of such UTQs is that they can be easily nested for scalable coding, which we discuss later in this chapter.

If we let the reconstruction levels be bin-wise conditional means and then entropy code this output, then UTQs are known to be close to optimal for common image probability density functions such as Gaussian and Laplacian. In order to control the generated bitrate via a buffer placed at output of the coder, a quality factor has been introduced. Both CBR and VBR strategies are of interest. We next describe the M-JPEG procedure.

M-JPEG Pseudo Algorithm

1. Input a new frame.
2. Scan to next 8×8 -image block.
3. Take 8×8 DCT of this block.

4. Quantize AC coefficients in each DCT block, making use of a *quantization matrix* $Q = \{Q(k_1, k_2)\}$,

$$\widehat{DCT}(k_1, k_2) = \text{Int} \left[\frac{DCT(k_1, k_2)}{s Q(k_1, k_2)} \right], \quad (12.1-1)$$

where s is a scale factor used to provide a rate control. As s increases, more coefficients will be quantized to zero and hence the bitrate will decrease. As s decreases toward zero, the bitrate will tend to go up. This scaling is inverted by multiplication by $s Q(k_1, k_2)$ at the receiver. The JPEG quality factor Q is inversely related to the scale factor s .

5. Form the difference of successive quantized DC terms ΔDC (effectively a noiseless spatial DPCM coding of the quantized DC terms).
6. Scan the 63 AC coefficients in a conventional zigzag scan.
7. Variable length code this zigzag sequence as follows:
 - a. Obtain run length (RL) to next nonzero symbol.
 - b. Code the pair (RL, nonzero symbol value) using a 2-D *Huffman* table.
8. Transmit (store, packetize) bitstream with end-of-block (EOB) markers. In addition to this coded data, a file header would contain information on quantization matrix Q , Huffman table, image size in pixels, color space used, bit depth, frame rate, etc.
9. If there are more data in frame, return to step 2.
10. If there are more frames in sequence, return to step 1.

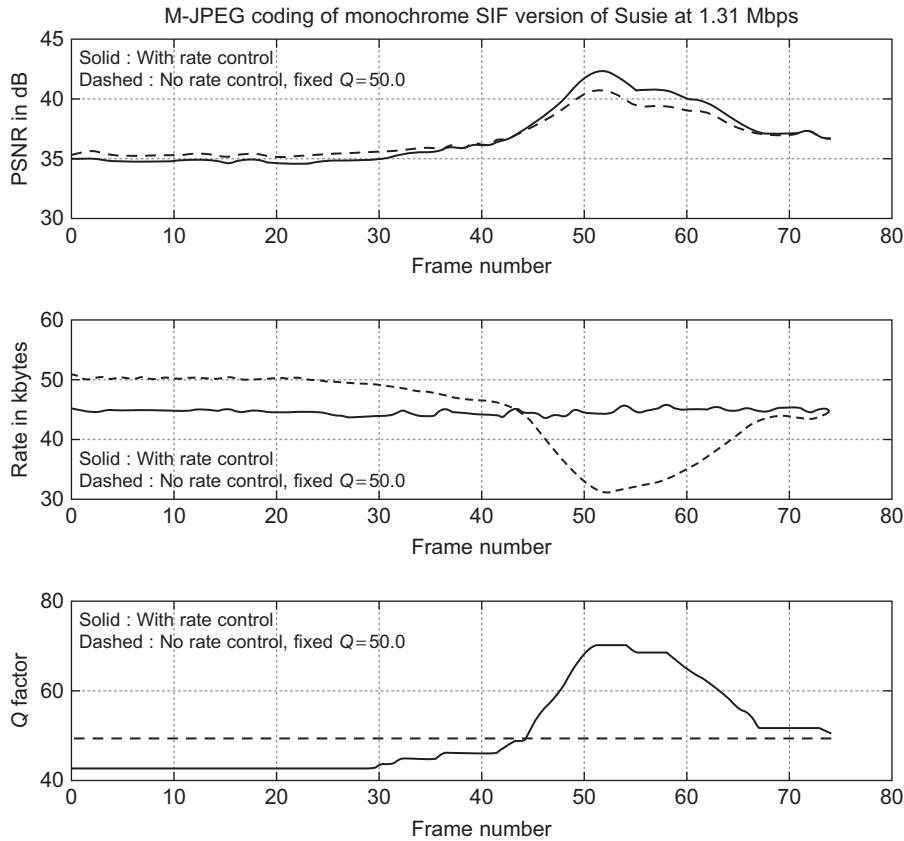
Variations on this M-JPEG algorithm have been used in both consumer and professional camcorders and video editing software. A key aspect needed when JPEG is extended to video is rate control, which calculates the scale factor s in this algorithm. The following example shows a simple type of *rate control* with the goal of achieving CBR on a frame-to-frame basis (i.e., intraframe rate control). It should be mentioned that while JPEG is an image coding standard, M-JPEG is not a video coding standard. So various flavors of M-JPEG exist and are not exactly compatible.

Example 12.1-1: M-JPEG

Here, we present two nearly constant bitrate examples obtained with a buffer and a simple feedback control to adjust the JPEG quality factor Q . Our control strategy is to first fit a straight line of slope γ to plot the Q factor versus $\log R$ of a sample frame in the clip to estimate an appropriate step size, and then employ a few iterations of steepest descent. Further, we use the Q factor of the prior frame as the first guess for the present frame. The update equation becomes

$$Q_{\text{new}} = Q_{\text{prev}} + \gamma (\log R_{\text{target}} - \log R_{\text{prev}}).$$

We ended up with 1.2–1.4 iterations per frame on average to produce the following results. We have done two SIF examples: *Susie* at 1.3 Mbps and *Table Tennis* at 2.3 Mbps.

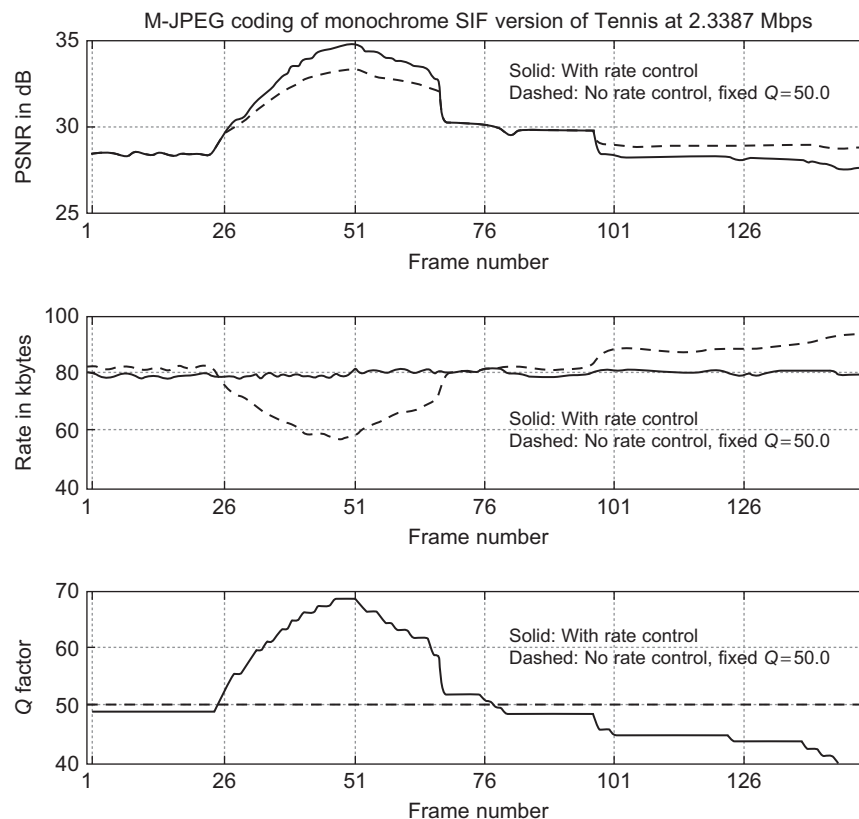
**FIGURE 12.1-3**

Rate control of M-JPEG on the Susie clip.

Figure 12.1-3 shows the results for the Susie clip, from top to bottom: peak signal-to-noise ratio (PSNR), rate in KB/frame, and Q factor. We see a big variation in rate without rate control but a fairly constant bitrate using our simple rate control algorithm. Results from the second example, Table Tennis, are shown in Figure 12.1-4 where we note that the rate is controlled to around 76 KB/frame quite well.

Notice that in both of these cases, the constraint of constant rate has given rise to perhaps excessive increases in PSNR in the easy-to-code frames that contain blurring due to fast camera and/or object motion.

Most SD video has been, and continues to be, interlaced. However, in the presence of fast motion, grouping of two fields into a single frame prior to 8×8 DCT transformation is not very efficient. As a result, when M-JPEG is used on interlaced

**FIGURE 12.1–4**

An illustration of rate control performance for M-JPEG coding of the Table Tennis clip.

SD video, usually each field is coded separately, and there is no provision for combining fields into frames, which would be more efficient in the case of low motion. The common DV coder, still based on DCT, has such an adaptive feature, giving it an efficiency edge over M-JPEG.

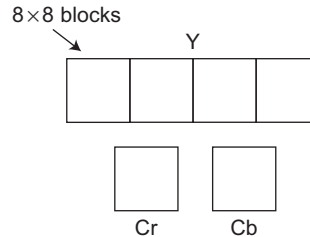
DV Codec

The source coder almost universally used in consumer SD camcorders is called *DV coder* [1], and there is some variation in the details of how they work. These intraframe coders first subsample the chroma components in the NTSC system in 4:1:1 fashion,¹ as shown in Figure 12.1–5, where the x symbols indicate luma

¹A PAL version is available that uses a 4:2:0 color space at 25 interlaced fps.

**FIGURE 12.1-5**

4:1:1 color subsampling pattern of DV in the so-called 525/30 NTSC system.

**FIGURE 12.1-6**

The DV macroblock for the NTSC (525/30) system.

samples and the o symbols represent chroma (both *Cr* and *Cb*) sample locations in a field. Note that the picture sites of the two chroma samples occur at a luma sample site. This is not always the case for 4:1:1 but is true for NTSC DV coders.

There is a provision for combining the two fields of the interlaced NTSC system into a frame or not, depending on a mode decision (not standardized, but left up to the DV coder designer). Then 8×8 DCTs of either fields or frames are performed and grouped into *macroblocks* as shown in Figure 12.1-6.

Thus a DV macroblock consists of six blocks: four luma and two chroma blocks. These macroblocks are then quantized and variable-length coded. But first, they are mapped one-to-one onto *sync blocks* and then stored K at a time into *segments*. Each segment is assigned a fixed code length in order to permit robust reading of digital tape at high speed, for so-called “trick play,” like search in fast-forward. Thus VLC is only used in DV coders within segments. The quantization is performed similar to (12.1-1), with the scaling variable s controlled to achieve a fixed target bitrate per segment. In what is called a *feed-forward* or *look-ahead strategy*, 8 to 16 different values of s are tried within the segment to meet the goal for fixed bit assignment. The best one is then selected for the actual coding of the segment. The DV standard adopted $K = 30$ macroblocks per segment, and they are distributed “randomly” around the image frame to improve the uniformity of picture quality, which at 25 Mbps is generally between 35 and 40 dB for typical interlaced SD video content. The DV standard for SD video has been largely supplanted by HDV introduced in 2003 as a consumer format for HD video. It is not an intraframe coder, but rather is based on the interframe coder MPEG 2, to be discussed in Section 12.3.

Both the M-JPEG and DV coder have a constant bitrate per frame and so are CBR. This means that the video quality will be variable, depending on the difficulty of the

various image frames. They effectively are buffered at the frame level to achieve this CBR property. Use of a longer buffer would allow VBR operation, wherein we could approach constant quality per frame.

Intraframe SWT Coding

Here, we look at intraframe video coders, which use SWT in place of DCT. Again, the need for some kind of control of the video bitrate is the new main issue over SWT image coding. First, we consider the rate assignment over the spatial subbands within each frame.

We must split up the total bits for each frame into the bit assignment for each subband. If we assume an orthogonal analysis/synthesis SWT, then the total mean-square error distortion D due to the individual quantizations, with assumed distortion $D_m(R_m)$ for R_m bits assigned to the m th subband quantizer, can be written as the weighted sum of the mean-square distortions over the M subbands as

$$D = \sum_{m=1}^M \frac{N_m}{N} D_m(R_m), \quad (12.1-2)$$

as pointed out for the image compression case in Chapter 9, where N_m is the number of samples in the m th subband, and N is the total number of samples. Then modeling the individual quantizers as $D_m = g\sigma_m^2 2^{-2R_m}$, all with the same quantizer efficiency factor g , the optimal bit assignment for an average bitrate of R bits/pixel, was determined as

$$R_m = R + \frac{1}{2} \log_2 \left(\frac{\sigma_m^2}{\sigma_{\text{wgm}}^2} \right), \quad m = 1, \dots, M, \quad (12.1-3)$$

where the weighted geometric mean σ_{wgm}^2 is defined as

$$\sigma_{\text{wgm}}^2 \triangleq \prod_{m=1}^M (\sigma_m^2)^{N_m/N}.$$

The resulting MSE per frame in the reconstruction then becomes

$$D = g\sigma_{\text{wgm}}^2 2^{-2R},$$

which is analogous to the case of SWT image coding (see Chapter 9).

If we choose to have CBR, then this is the solution. However, we may want to either achieve constant distortion or minimum average distortion. This then requires that we vary the bit assignment over the frames. Given the image frames' subband variances, we can compute the weighted geometric mean of each frame $\sigma_{\text{wgm}}^2(n)$; the

distortion at frame $D(n)$ will then be

$$D(n) = g\sigma_{\text{wgm}}^2(n)2^{-2R(n)},$$

so that the total distortion over all the frames becomes

$$\begin{aligned} D_T &= \sum_{n=1}^N D(n) \\ &= \sum_{n=1}^N g\sigma_{\text{wgm}}^2(n)2^{-2R(n)}, \end{aligned} \quad (12.1-4)$$

where $R(n)$ is the bit assignment to the n th frame, and the average bitrate is then

$$\frac{1}{N}R_T = \frac{1}{N} \sum_{n=1}^N R(n). \quad (12.1-5)$$

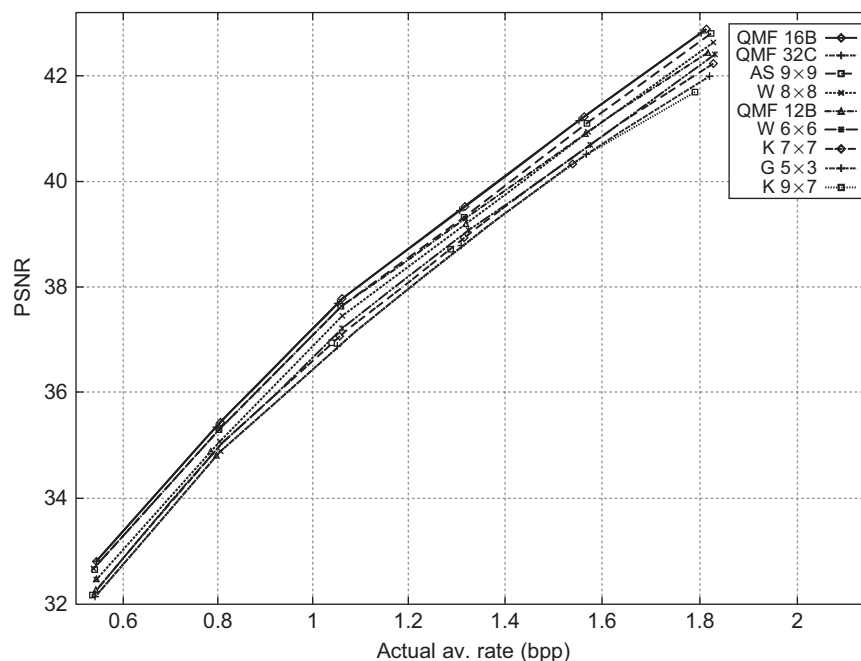
This pair of operational rate and distortion functions (12.1-4) and (12.1-5) can then be optimized for best average performance over time. Normally this minimum will not occur at the point where all the frames have the same distortion $D(n) = D$. In that case, one has to make the choice of either optimizing the constant distortion performance or optimizing for best average performance. They are generally different.

One issue in SWT coding is the choice of the subband/wavelet filter to use, as considered in the following example.

Example 12.1-2: SWT Filter Study

The goal is to see the effect of different SWT filters on an efficient intraframe coder. Individual frames are first subband analyzed and then individually quantized by UTQs. The output of the quantizers is then separately arithmetic coded. The closed form log bit-assignment rule of this section was used. We also used the arithmetic coding algorithm of [2]. Generalized Gaussian models, whose parameters were estimated from first frame, were employed. The AC output was grouped into sync blocks: LL-LL subband blocked to two lines, LL-LH, -HL, -HH subbands blocked to four lines, and the higher subbands not blocked. A 5-bit end-of-block marker was used to separate these sync blocks. The results from [3] for the HD test clip *MIT sequence* are shown in Figure 12.1-7.

We see that there is not a drastic PSNR difference in the performance of these filters, but here the longer QMFs do better. The popular CDF 9/7 wavelet filter was not included in the study but was later found to have similar performance, slightly better

**FIGURE 12.1-7**

Comparison of various SWT filters for intraframe video compression.

than the QMFs for dyadic or wavelet decomposition, but slightly worse for full subband decomposition [4].

M-JPEG 2000

M-JPEG 2000 is officially JPEG 2000, Part 3, and amounts to a standardized file format or container for a sequence of JPEG 2000 image files. It can be CBR or VBR within the standard. Various methods of bitrate control can be applied to the resulting sequence of coded frames, making them either constant quality, maximum average quality, HVS weighted quality, etc. The M-JPEG 2000 standard is just the standard for the decoder, thus permitting any variable numbers of bits per frame. As always, rate control is only an encoder problem.

As mentioned in Chapter 9, JPEG 2000 uses operational rate-distortion optimization over tiles on the image frame. These tiles are then coded using the *constant slope condition*, resulting in minimum average mean-square coding error. This constant slope matching of JPEG 2000 can be extended between frames to achieve a minimum mean-square coding error for the whole image sequence. Such an approach may mean an unacceptable delay, so a matching of slope points on a local frame-by-frame

basis may be chosen as an alternative VBR solution that has been found much closer to constant quality than is CBR.

Example 12.1–3: Digital Cinema

In 2004, Digital Cinema Initiatives (DCI) ran a test of compression algorithms for the digital cinema distribution at both 4K and 2K resolution, meaning horizontal number of pixels and aspect ratios around 2.35. DCI had been created 2 years earlier by the major U.S. motion picture companies. A test suite was generated for the tests by the American Society of Cinematographers (ASC) and consisted of 4K RGB image frames at 24 fps. The bit depth of each component was 12 bits. After extensive testing, involving both expert and nonexpert viewers, DCI made the recommendation of M-JPEG 2000 as the best choice for digital cinema. In part this was because of the 4K/2K scalability property, and in part because of its excellent compression performance. The tests used the CDF 9/7 filter or lossy version of the M-JPEG 2000 coder. The JPEG committee later standardized a new profile of JPEG 2000 Part 1 specially for digital cinema use. Bitrates are in the vicinity of 100–200 Mbps at the 4K resolution.

We now turn to the generally more efficient interframe coders. While these coders offer more compression efficiency than the preceding intraframe coders, they are not so artifact-free and are mainly used for *distribution quality* purposes. Intraframe coders, with the exception of the digital cinema distribution standard, have been mainly used for *contribution quality* (i.e., professional applications) because of their high quality at moderate bitrates and their ease of editing.

12.2 INTERFRAME CODING

Now we look at coders that make use of the dependence between or *inter* frame. We consider spatiotemporal (3-D) DPCM, basic MC-DCT concepts, MPEG/VCEG video coding, H.26x visual conferencing, and MC-SWT coding. We start with the 1-D DPCM coder and generalize it to encode entire image frames in the temporal direction.

Generalizing 1-D DPCM to Interframe Coding

We replace the 1-D scalar value $x(n)$ with the video frame $x(n_1, n_2, n)$ and do the prediction from a *nonsymmetric half-space* (NSHS) region. However, in practice the prediction is usually based only on the prior frame or frames. Then we quantize the prediction error difference or residual. We thus have the system shown in [Figure 12.2–1](#). We can perform *conditional replenishment* [5], which only transmits pixel significant differences (i.e., beyond a certain threshold). These significant differences tend to be clustered in the frame so that we can efficiently transmit them

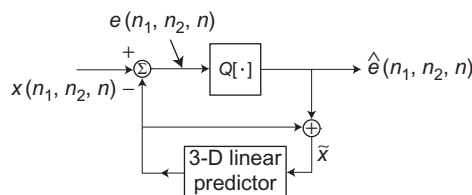


FIGURE 12.2–1

Spatiotemporal generalization of DPCM with spatiotemporal predictor.

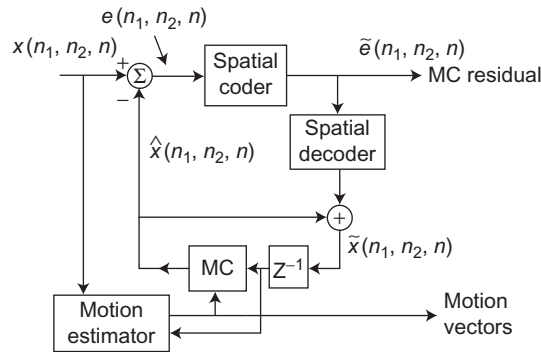
with a context-adaptive VLC. The average bitrate at buffer output can be controlled by a variable threshold for significant differences. For an excellent summary of this and other early contributions, see the 1980 review article by Netravali and Limb [6].

Looking at Figure 12.2–1, we can generalize it by putting any 2-D spatial or intraframe coder in place of the scalar quantizer $Q[\cdot]$ in the generalized DPCM. If we use block DCT for the spatial coder block that replaced the quantizer, we have a *hybrid coder*, being temporally DPCM and spatially transform based. We can use a frame-based predictor, whose most general case would be a 3-D nonlinear predictor operating on a number of past frames. Often, though, just a frame delay is used, effectively predicting that the current frame will be the same as the motion-warped version of one previous frame. In some current coding standards, a spatial filter (called a *loop filter* since it is inside the feedback loop) is used to shape the quantizer noise spectrum and to add temporal stability to this otherwise only marginally stable system. Another way to stabilize the DPCM loop is to put in an *Intra* frame every so often. Calling such frame an *I* frame, and the intercoded frames *P*, we can denote this as *IPPP ... PIPPP ... PIPPP ...*. Another idea, in the same context, would be to randomly insert *I* blocks instead of complete *I* frames. While the former refresh method is used in the MPEG 2 entertainment standard, the latter method is used in the H.263 visual conferencing standard.

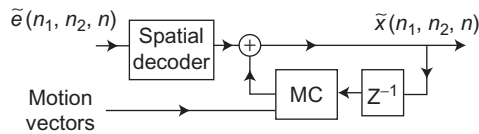
MC Spatiotemporal Prediction

There are two types of motion-compensated hybrid coders. They differ in the kind of motion estimation they employ: *forward* motion estimation or *backward* motion estimation. The MC block in Figure 12.2–2 performs the motion compensation (warping) after the motion estimation block computes the forward motion. The quantity $\tilde{x}(n_1, n_2, n)$ seen in the encoder is also the decoded output; that is, the encoder contains a decoder. The actual decoder is shown in Figure 12.2–3; it consists of first a spatial decoder followed by the familiar DPCM temporal decoding loop, as modified by the motion compensation warping operation MC that is controlled by the received motion vectors.

In *forward MC*, motion vectors are estimated between the current input frame and the prior decoded frame. Then the motion vectors must be coded and sent along with the MC residual data as side information since the decoder does not have access to the input frame $x(n_1, n_2, n)$. In *backward MC*, we base the motion estimation on the previous two decoded frames $\tilde{x}(n_1, n_2, n-1)$ and $\tilde{x}(n_1, n_2, n-2)$, as shown in

**FIGURE 12.2-2**

Illustrative system diagram for motion-compensated DPCM.

**FIGURE 12.2-3**

The hybrid decoder.

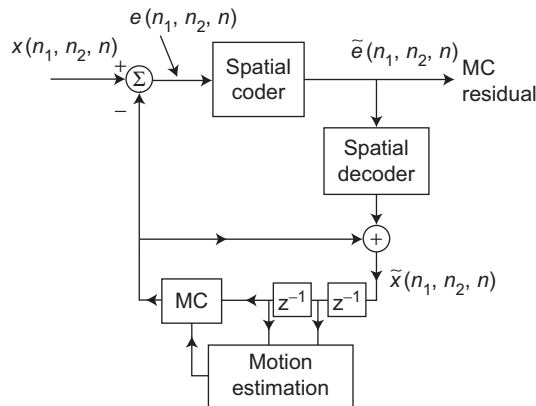
**FIGURE 12.2-4**

Illustration of backward motion compensation.

Figure 12.2-4. Then there is no need to transmit motion vectors as side information, since in the absence of channel errors the decoder can perform the same calculation as the encoder and come up with the same motion vectors. Of course, from the viewpoint of total computation, backward MC means doing the computationally intensive motion estimation and compensation work twice, once at the coder and once

at the decoder. (You are asked to write the decoder for backward motion-compensated hybrid coding as an end-of-chapter problem.)

One question with interframe coding is the required accuracy in representing the motion vectors. There has been some theoretical work on this subject [7] that uses a very simple motion model. Even so, insight is gained on required motion vector accuracy, especially as regards the level of noise in the input frames.

12.3 EARLY INTERFRAME CODING STANDARDS

The MPEG committee is a part of the *International Standards Organization* (ISO) and introduced its first video compression standard for CD-based video in 1988. It was called MPEG 1. A few years later, they released a standard for SD video called MPEG 2. This standard has been widely used for distribution of digital video over satellite, DVD, terrestrial TV broadcast, and cable. It was later extended to HD video. These are really standards for decoders. Manufacturers are left free to create encoders with proprietary features that they feel improve picture quality, reduce needed bitrate, or improve functionality. The only requirement is that these coders create a bitstream that can be decoded by the standard decoder, thus ensuring interoperability. Since you cannot really create a decoder without an encoder, MPEG developed various test model coders, or *verification models*, some of which have become widely available on the Web. However, the performance of these “MPEG coders” varies and should not be considered to indicate the full capabilities of the standard. In the parlance of MPEG, these verification model coders are nonnormative (i.e., not part of the standard). The best available coders for an MPEG standard are usually proprietary.

MPEG 1

The first MPEG standard, MPEG 1, was intended for the SIF resolution, about half NTSC television resolution in each dimensions. Exactly, this is 352×240 luma pixels. The chroma resolution is 2×2 reduced by half again to 176×120 chroma pixels. Temporally, SIF has a 30-Hz frame rate, that is progressive or noninterlaced. The aspect ratio is 4:3. The MPEG 1 standard was meant for bitrates between 1 and 2 Mbit/sec.

The overall coding method was intended for digital video and audio on 1x CDs in the late 1980s with a maximum data rate of 1.5 Mbps, with 1.2 Mbps used for video. They wished to provide for fast search, fast backward search, easy still-frame display, limited error propagation, and support for fast image acquisition starting at an arbitrary point in the bitstream. These needed functionalities make it difficult to use conventional frame-difference coding—i.e., to use temporal DPCM directly on all the frames. The MPEG solution is to block a number of frames into *groups of pictures* (GOPs), and then to code these GOPs separately.

Each GOP must start with a frame that is intracoded, the *I* frame. This will limit efficiency, but facilitate random search, which can be conducted on the *I* frames. An illustration of GOP structure is shown in Figure 12.3–1. If the GOP is not too long, say half a second, then the rapid image acquisition functionality can be achieved too. After the initial *I* frame, the remaining frames in the GOP are interframe coded. MPEG has two types of intercoded frames, *B* bi-directional MC-interpolated and *P* frames that are MC-predicted. The typical GOP size in video is $N = 15$, with $I = 1$ intraframe, $U = 4$ predicted frames, and $B = 10$ interpolated frames. A parameter $M = 2$ here denotes the number of *B* frames between each *P* frame. After the *I* frame, we next predictively code the *P* frames one by one, as seen in Figure 12.3–2. After its neighboring *P* frames are coded, the bidirectional interpolation error of the *B* frames is calculated (and quantized), as illustrated in Figure 12.3–3. In MPEG parlance, with reference to this figure, the neighboring *I* and *P* frames serve as references for the coding of the *B* frames.

In MPEG 1, motion compensation is conducted by block matching on *macroblocks* of size 16×16 . Exactly how to do the block matching is left open in the standard, as well as the size of the search area. Accuracy of the motion vectors was specified at the integer pixel level, sometimes called “full pixel accuracy,” the accuracy naturally arising from block-matching motion estimation. In fact, most MPEG 1 coders use a fast search method in their block matching.

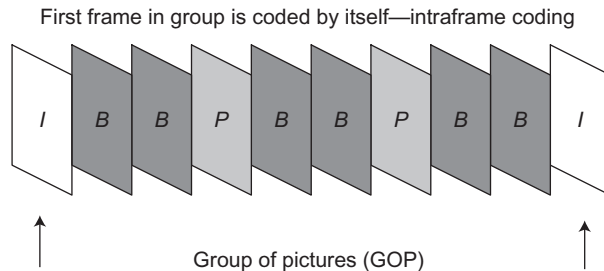


FIGURE 12.3–1

First code *I* frames (N frames in group).

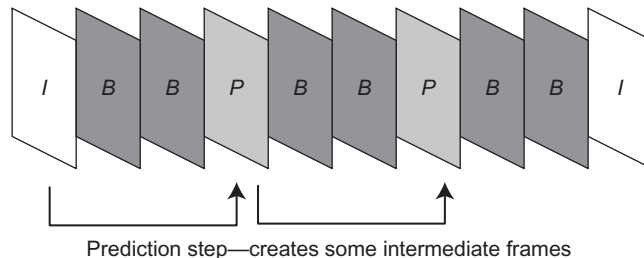


FIGURE 12.3–2

Illustration of predictive coding of *P* frames in MPEG 1.

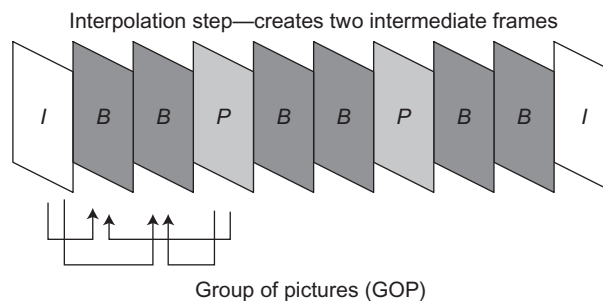
**FIGURE 12.3–3**

Illustration of coding the *B* frames based on bidirectional references.

We can see that MPEG 1 uses a layered coding approach. At the highest level is the GOP layer. The GOP in turn is made up of *pictures I, P, and B*. Each picture is composed of an array of macroblocks. Each macroblock consists of a 2×2 array of 8×8 blocks. The DCT is performed at this block level. There is one additional level, the *slice* level, that occurs within a picture. A slice is just a row or contiguous set of rows of macroblocks. Its main purpose is synchronization in the face of channel errors or lost packets in transmission. VLC is performed only within a slice, and every slice is terminated with a unique *end of slice* (EOS) codeword. When this EOS codeword is detected in the sequence, end of slice is declared, and any remnant effects of past channel errors terminate. Another advantage of partitioning the coded data into slices is to provide more decoder access points, since there is usually a slice header that specifies frame number and position information. Usually cross-references across slices are disallowed, thus incurring at least a slight bitrate efficiency penalty for using small slices.

MPEG 1 achieves CBR by employing a so-called *video buffer verifier* in the coder. It serves as a stand in for the buffer that must be used at a decoder, which it is fed by a constant bitrate from the channel. By keeping the video buffer verifier from underflow or overflow during coding, one can guarantee that there will be no buffer problems during decoding.

MPEG 2—A “Generic Standard”

The MPEG2 coder was developed jointly with the VCEG committee of the International Telecommunications Union (ITU) and is also known as H.262 [8]. It includes MPEG 1 as subset and does this by introducing *profiles* and *levels*. The level refers to the image size (i.e., CIF, SIF, or SD), frame rate, and whether the data are interlaced or progressive. The profile refers to the MPEG 2 features that are used in the coding. MPEG 2 also established new prediction modes for interlaced SD, permitting prediction to be *field based* or *frame based*, or a combination of the two. The predictions and interpolations had to consider the interlaced nature of common SD NTSC and PAL data. As a result, various combinations of field-based and frame-based prediction and interpolation modes are used in MPEG 2. For example, when bidirectionally predicting (interpolating) the *B* frames in Figure 12.3–4, there are both field-based

and frame-based prediction modes. In field-based mode, which usually works best for fast motion, each field is predicted and each difference field is coded separately. In frame-based mode, which works best under conditions of slow or no motion, the frame is treated as a whole and coded as a unit. In field-based mode there is the question of what data to use as the reference field—i.e., which field, *upper* or *lower*.² The additional overhead information of the prediction mode, field or frame, has to be coded and sent along as overhead. This provision for interlaced video makes MPEG 2 more complicated than MPEG 1, which only handled progressive video, but reportedly gives it about a 0.5 to 2.0 dB advantage [9] versus frame-based mode alone. This is due to the combined and adaptive use of frame/field prediction and frame/field DCT. Additionally, whether interlaced or progressive, MPEG 2 permits use of half-pixel accurate motion vectors for improved prediction and interpolation accuracy at the cost of higher motion vector rate. Most all MPEG 2 coders make use of this feature.

Commonly used color spaces in MPEG 2 are 4:2:2 for professional applications and 4:2:0 for consumer delivery, such as on DVD or digital satellite television. The corresponding subsampling structures are shown in parts a and b, respectively, in Figure 12.3–5 [8].

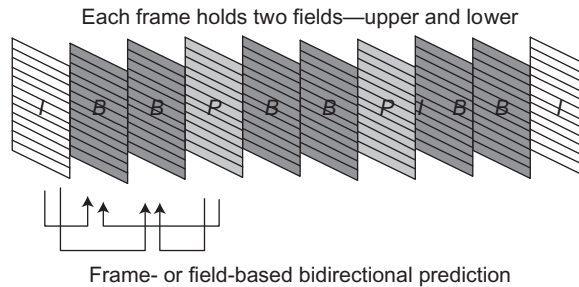


FIGURE 12.3–4

New in MPEG2 was the need to process interlaced frame data.

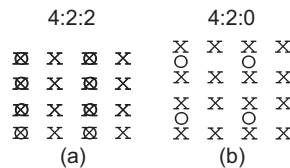


FIGURE 12.3–5

MPEG 2 4:2:2 (left) and 4:2:0 (right) luma and chroma subsampling sites.

²In MPEG 2, all data are stored in frames, and the field and frame numbers in a GOP are numbered starting with 0. Each frame consists of an upper or *even field*, which comes first, and a lower or *odd field*, which comes second.

Example 12.3–1: MPEG 2 Example

We coded the ITU 601 or SD test sequence Susie with a standard MPEG 2 coder available on the Web. The bitrate was 3 Mbps at a frame rate of 30 fps. We first used all *I* frames (i.e., GOP = 1), then we set GOP = 15 or a half second, and coded without *B* frames, and then with *B* frames. The PSNR results are shown in the Table 12.3–1. We see a jump in PSNR going from *I* frame only to *I* and *P*, showing the efficiency of the *P* frame in a hybrid coder. However, not much objective improvement for adding on the *B* frame is seen in this visual conferencing test clip. Videos are available for download at the book’s Web site.

Table 12.3–1 Some MPEG 2 Coding Results at 3 Mbps for the Susie SD Test Clip

PSNR	Y(dB)	U(dB)	V(dB)
<i>I</i>	33.8	44.7	44.6
<i>I,P</i>	39.0	45.7	45.5
<i>I,B,P</i>	39.1	46.1	45.8

A recent paper reported optimization of an MPEG 2 coder for IPPP... mode, indicating a 2 to 3 dB advantage [10] at high bitrates for the first 101 frames of quarter CIF (QCIF) Foreman versus the coding result of the widely used MPEG 2 test model 5 (TM5) [11] software.

The Missing MPEG 3—High-Definition Television

Standard definition video spatial resolution is not really adequate, so researchers worked for many years on an increased resolution system. Reading from report 801–4 of CCIR (now ITU) in 1990 they defined HDTV as “a system designed to allow viewing at about three times picture height, so that the system is virtually transparent to the quality of portrayal that would have been perceived in the original scene or performance by a discerning viewer with normal visual acuity.”

Researchers at NHK in Japan started modern research on HDTV in the early 1970s [12]. They came up with an analog system called *Hi-Vision* that enjoyed some deployment, mainly in Japan, but the equipment was very bulky, heavy, and expensive. This spurred U.S. companies and researchers to try to invent an all-digital approach in the mid- to late-1980s. A set of competing digital HDTV proposals then emerged in the United States. The Grand Alliance was formed in May 1993 from these proponents and offered a solution, eventually based on extending MPEG 2. It covered both interlaced and progressive scanning formats in a 16:9 aspect ratio (near to the 2:1 and higher aspect ratio of motion pictures). The resulting digital TV standard from the group now known as the Advanced Television Systems Committee (ATSC) offered 1080i and 720p digital video for terrestrial broadcast in

North America. More information on these formats can be found in the appendix at the end of Chapter 11. Since ATSC essentially voted in favor of MPEG 2, the HD and SD resolutions were essentially folded together into one coding family, just extending the profiles and levels of MPEG 2. Thus HD is coded with MPEG 2—main profile, at High level (MP@HL). As mentioned earlier, HDV camcorders also use MPEG 2 at HL.

MPEG 4—Natural and Synthetic Combined

The original concept of MPEG 4 was to find a much more efficient coder at low bitrates (e.g., 64 Kbps). When this target proved elusive, the effort turned to object-based video coding. In MPEG 4, so-called video objects (VO) are the basic elements to be coded. Their boundaries are coded by *shape coders* and their interiors are coded by so-called *texture coders*. The method was eventually extended up to SD and HD and is backward compatible with MPEG 2. A *synthetic natural hybrid coder* (SNHC) is capable of combining natural and synthetic (cartoon, graphics, text) together in a common frame. Nevertheless, most widely available embodiments of MPEG 4 work only with rectangular objects and one object per frame. Segmenting natural video into reasonable objects remains a hard problem.

Also starting with MPEG 4, there was an effort to separate the video coding from the actual transport bitstreams. They wanted MPEG 4 video to be carried on many different networks with different packet or cell sizes and qualities of service. So MPEG 4 simply provides a *delivery multimedia integration framework* for adapting the output of the MPEG 4 video coder, the so-called *elementary streams* (ES) to any of several already existing transport formats, such as MPEG 2 *transport stream* (TS), ATM, and IP networks (RTP/IP). Complete information on MPEG 4 is available in the resource book by Pereira and Ebrahimi [13].

Video Processing of MPEG-coded Bitstreams

Various video processing tasks have been investigated for coded data and MPEG2 in particular. In *video editing* of MPEG bitstreams, the question is how to take two or more MPEG input bitstreams and make one composite MPEG output stream. The problem with decoding/recoding is that it introduces additional coding noise and artifacts and is computationally demanding. Staying as much in the MPEG2 compressed domain as possible and reusing the editing mode decisions and motion vectors has been found essential. Since the GOP boundaries may not align, it is necessary to recode the one GOP where the edit point lies. Even then original bitrates may make the output *video buffer verifier* overflow. The solution is to requantize this GOP and perhaps neighboring GOPs to reduce the likelihood of such buffer overflow. The introduction of the HDV format, featuring MPEG 2 compression of HD video in camera, brought the MPEG bitstream editing problem to the forefront. Several software products are available that promise to edit HDV video in close to *native mode*.

The *transcoding of MPEG* question is how to go from a high quality level of MPEG to a lower one, without decoding and recoding at the desired output rate.

Transcoding is of interest for video-on-demand (VOD) applications. Transcoding is also of interest in video production work where *IPIP* ... may be used internally for editing and program creation, while the longer *IBBPBBP* ... GOP structure is necessary for program distribution. Finally, there is the worldwide distribution problem, where 525- and 625-line systems³ continue to coexist. Here, a motion-compensated transcoding of the MPEG bitstream is required, due to the differing 60 and 50 Hz frame rates. Current HD formats have been standardized at 1080i and 720p, but 60 and 50 Hz frame rate variants continue to coexist. For more on transcoding, see Section 6.3 in Bovik's handbook [14].

H.263 Coder for Visual Conference

This coder from the VCEG group at ITU evolved from their earlier H.261 or px64 coder. As the original name implies, it is targeted at rates that are a multiple of 64 Kb/sec. To achieve such low bitrates, they resort to a small QCIF frame size and a variable and low frame rate, with bitrate control based on buffer fullness. If there is a lot of motion in detailed areas that generate a lot of bits, then the buffer fills and the frame rate is reduced (i.e., frames are dropped at the encoder). The user can specify a target frame rate, but often the H.263 coder at, say 64 Kbps will not achieve a target frame rate of 10 fps. The H.263 coder features a *group of blocks* (GOB) structure, rather than a GOP, with *I* blocks being inserted randomly for refresh. While there are no *B* frames, there is the option for so-called *PB* frames. The coder has half-pixel accurate motion vectors like MPEG 2 and can use overlapped motion vectors from neighboring blocks to achieve a smoother motion field and reduced blocking artifacts. Also there is an advanced prediction mode option and an arithmetic coder option.

The reason for targeting the GOB structure versus the GOP structure is the need to avoid the *I* frames in a GOP structure that require a lot of bits to transmit, a difficulty in video phone that H.263 targeted as a main application. In video phone, low bitrates and a short latency requirement (≤ 200 msec) mitigate the bit-hungry *I* frames. As a result, in H.263, slices or GOBs are updated by *I* slices randomly, thus reducing the variance on coded frame sizes that occurs with the GOP structure. High variance of bits/frame was not a problem in MPEG 2 because of its targeted entertainment applications like video streaming, including multicasting, digital broadcasting, and DVD. Some low bitrate H.263 coding results are available at the book's Web site. Current interframe coding standards are covered in [Section 12.6](#).

12.4 INTERFRAME SWT CODERS

The first 3-D subband coding was done by Karlson [15] in 1989. This first 3-D subband coding used Haar or two-tap subband/wavelet filters in the temporal direction

³The reader should note that the 525-line system only has 486 visible lines, meaning it is SD, which is 720×486 . A similar statement is true for the 625-line system. The remaining lines are hidden!

and separable LeGall/Tabatabai (LGT) 5/3 spatial filters [16], whose 1-D component filters are

$$\begin{aligned} H_0(z) &= \frac{1}{4}(-1 + 2z^{-1} + 6z^{-2} + 2z^{-3} - z^{-4}), \\ H_1(z) &= \frac{1}{4}(1 - 2z^{-1} + z^{-2}), \\ G_0(z) &= \frac{1}{4}(1 + 2z^{-1} + z^{-2}), \text{ and} \\ G_1(z) &= \frac{1}{4}(1 + 2z^{-1} - 6z^{-2} + 2z^{-3} + z^{-4}). \end{aligned}$$

The system was subjectively optimized using scalar quantization, and DPCM to increase coding efficiency for the base subband. The 3-D frequency domain subband division of Karlson is shown in Figure 12.4-1 and achieves the 3-D frequency decomposition shown in Figure 12.4-2.

A simple 3-D subband coder was also designed by Glenn et al. [17], which used perceptual coding based on the human contrast sensitivity function but like the Karlson coder, no motion compensation. A real-time demonstration hardware offered 30:1 compression of SD video up to a bitrate of 6 Mbits/sec. The visual performance was said to be comparable to MPEG 2, in that it traded slight MPEG artifacts for this coder's slight softening of moving scenes. Such softening of moving objects is typical of 3-D transform coders that do not use motion compensation.

Motion-Compensated SWT Hybrid Coding

Naveen and Woods [18, 19] produced a hybrid SWT coder with a multiresolution feature. Both pel-recursion in the context of backward motion estimation and block

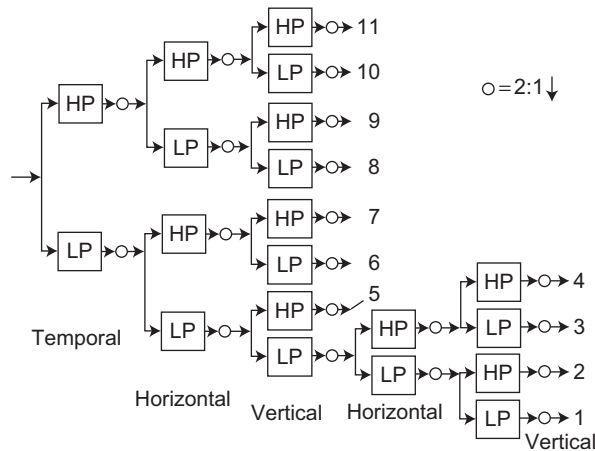
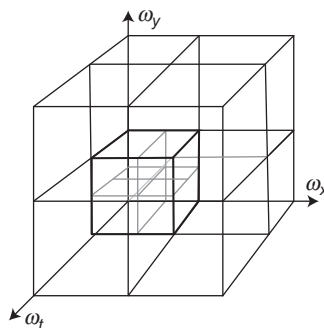


FIGURE 12.4-1

First 3-D subband filter, due to Karlson.

**FIGURE 12.4-2**

Frequency decomposition of Karlson's filter tree.

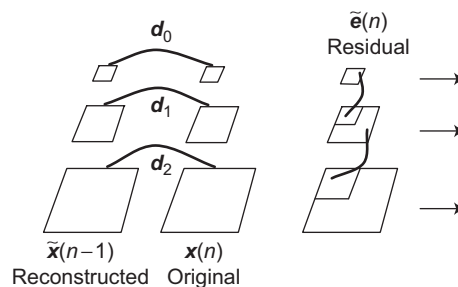
**FIGURE 12.4-3**

Illustration of the multiresolution coder.

matching for forward motion estimation were considered. They used a subband pyramid for hierarchical motion estimation for the multiresolution transmission of HD and SD video. Their basic motion compensation approach placed the motion compensation inside the hierarchical coding loop to avoid drift. There is an implicit assumption here that the LL subband of the full or HD resolution is a suitable candidate for the SD video.

An illustration of their motion estimation pyramid and coding pyramid is given in Figure 12.4-3, which shows two subband/wavelet pyramids on the left for the previous reconstructed frame $\tilde{x}(n-1)$ and, in the middle, the current input frame $x(n)$. The prediction error residual, to be coded and transmitted, $e(n)$, is shown on the right. First, hierarchical block matching determines a set of displacement vector fields d_0, d_1 , and d_2 of increasing resolution. A prediction residual is then created for the lowest resolution level and is coded and transmitted. The corresponding low resolution reconstructed value is then used to predict the LL subband of the next higher resolution level. For this level, the prediction error is transmitted for the LL subband, while the coded residuals are transmitted for the higher resolutions. The process is repeated for the third spatial resolution. In this scheme, there arises the need to make a bit assignment for the various levels to try to meet given quality requirements.

In the context of block-matching motion estimation with SWT coding of the residual, the motion vector discontinuities at the block boundaries can increase the energy in the high-frequency coefficients. This situation does not arise in DCT coding because the DCT block size is kept at a subpartition size with respect to the motion vector blocks; for example, common DCT size 8×8 with common motion vector block sizes are 16×16 and 8×8 in MPEG 2. Thus the DCT transform does not see the artificial edge in the prediction residual frame caused by the motion vector discontinuity at the motion vector block boundaries. Quite the contrary for the SWT transform where there are no block boundaries, so that any discontinuities caused by the block motion vector prediction are a serious problem. To counter this, a small amount of smoothing of motion vectors at the block boundaries was used in [19] for the case of forward motion compensation. The problem does not arise in the backward motion compensation case where a dense motion estimate can be used, such as the pel-recursive method, since the motion field does not have to be transmitted.

Gharavi [20] used block matching and achieved integer pixel accuracy in subbands permitting the parallel processing of each subband at four times the lower sample rate. However, Vandendorp [21] pointed out that neighboring subbands must be used for *inband* (in-subband) *motion compensation*. Bosveld [22] continued the investigation and found MSE prediction gains almost equal to those in the spatial domain case. The result is more embedded data sets that facilitate scalability. Usually just the nearest or bordering subbands contribute significantly to in-band values.

3-D or Spatiotemporal Transform Coding

Kronander [23] took the temporal DCT in blocks of four or eight frames and motion compensated then towards a central frame in the block. A spatial subband/wavelet filter was then used to code the MC residual frames in the block. Decoding proceeded by first decoding the residual frames and then inverting the MC process.

Consider a block of compressed images available at time k , and denote the l th such image

$$\mathbf{y}_{k,l} = \mathbf{x}_{k,l}(\mathbf{t} - \mathbf{v}_{k,l}(\mathbf{t}) \triangle t) ; l = 0, \dots, L-1,$$

where \mathbf{t} is a spatial position with velocity $\mathbf{v}_{k,l}$ to some central reference frame l' . We then perform spatial subband/wavelet coding/decoding of the temporal prediction residue,

$$\mathbf{y}_{k,l} \rightarrow \tilde{\mathbf{y}}_{k,l}.$$

Finally, we approximately invert the displacement information to obtain the decoded frames,

$$\tilde{\mathbf{x}}_{k,l}(\mathbf{t}) = \tilde{\mathbf{y}}_{k,l}(\mathbf{t} - \mathbf{u}_{k,l}(\mathbf{t}) \triangle t),$$

where $\mathbf{u}_{k,l}$ is the inverse to $\mathbf{v}_{k,l}$, at least approximately, and then proceed to the next block, $k+1$.

Conditions for invertibility, simply stated, are “if the transformation were painted on a rubber sheet... a good compensation algorithm should distort the rubber but not fold it” [23]. This effectively precludes covered/uncovered regions. The motion estimation methods used were full-search block matching with a 8×8 block and 16×16 search area, and hierarchical block matching with a 7×7 block and 3×3 search at each level. Motion was inverted by simple negation of displacement vectors.

Ohm’s SWT coder MC-SBC [24] featured a *motion-compensated temporal filter* (MCTF), a type of 3-D subband/wavelet filter that attempts to filter along the local motion paths. For the temporal filtering, Ohm made use of the two-tap Haar transform. Due to the fact that MCTF is nonrecursive in time, we can expect limited error propagation and importantly no temporal drift in scalable applications. A four-temporal level MCTF with $\text{GOP} = 16$ is shown in Figure 12.4-4. This MCTF uses the Haar filter, with motion fields between successive pairs of frames indicated with curved arcs. Proceeding down the diagram, we go to lower temporal levels or frame rates by a factor of 2 each time. The lowest frame rate here is 1/16th of the full frame rate. The numbers in parentheses denote the temporal levels, with 1 being the lowest and 5, or full frame rate, being the highest. Note that the motion vectors are numbered with the temporal levels that they serve to reconstruct. In Figure 12.4-4, we see motion vectors represented as rightward arrows, terminating at the location of $t - H$ frames. Different with the now somewhat old forward and backward MC terminology used in the hybrid coding, in MCTF parlance, this is called *forward motion compensation*, with backward motion compensation being reserved for the case where the arrows point to the left. In both cases, the motion vectors must be transmitted.

A corresponding three-level diagram for the LGT 5/3 filter is shown in Figure 12.4-5, where we notice that twice as much motion information is needed

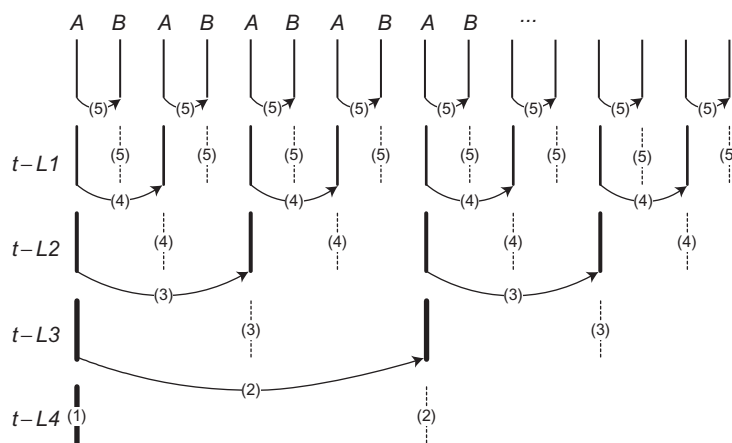


FIGURE 12.4-4

Illustration of four-level Haar filter MCTF.

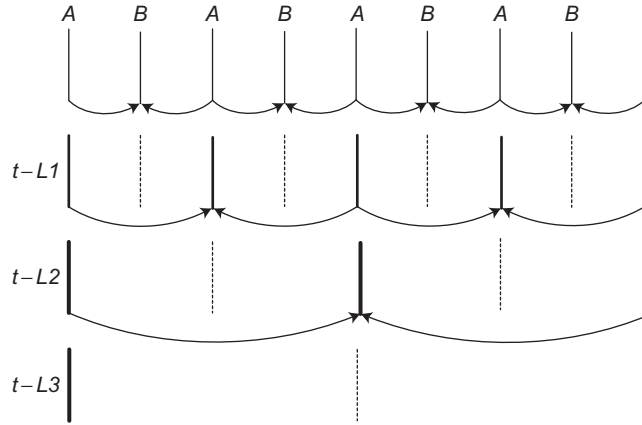
**FIGURE 12.4-5**

Illustration of LGT 5/3 filter MCTF.

as in the Haar filter MCTF. Also note that the GOP concept is blurred by the references going outside the set of eight frames. So for 5/3 and longer SWT filter sets, it is better to talk of levels of the MCTF than the GOP size.

In equations, we can write the lifting steps, at each temporal level, for each of these cases as follows:

Haar case:

$$h(n) = x_A(n) - x_B(n - d_f), \quad (12.4-1)$$

$$l(n) = x_B(n) + \frac{1}{2}h(n + d_f). \quad (12.4-2)$$

LGT 5/3 case:

$$h(n) = x_A(n) - \frac{1}{2} [x_{B_f}(n - d_f) + x_{B_r}(n - d_r)], \quad (12.4-3)$$

$$l(n) = x_B(n) + \frac{1}{4} [h_f(n + d_f) + h_r(n + d_r)]. \quad (12.4-4)$$

In these equations, we have labeled the frames in terms of A and B , as shown in Figures 12.4-4 and 12.4-5, and also used d_f and d_r to refer to the forward and backward motion vectors, from frame A to reference frames B on either side (i.e., B_f and B_r), where f stands for *forward* and r stands for *reverse*. These operations are then repeated at the second and subsequent levels of decomposition. We also notice that backward or reverse motion vectors (i.e., $d = -d_f$) are used in the update step for the LGT 5/3 lifting implementation. (This is also done in the Haar case.) Finally, in the subpixel accurate case, many of the terms in these two equations must be computed by interpolation. The beauty of the lifting implementation, however, is that this is

exactly invertible in the corresponding SWT synthesis operation, so long as the same interpolation method is used.

Example 12.4–1: Haar MCTF with Four Levels

With respect to Figure 12.4–4, note that frame $t - L4$ is an intraframe to be coded and sent every 16 frames. Hence there is a natural temporal block size of 16 here in this four-level Haar case. So we say $\text{GOP} = 16$ here. The $t - L4$ frame subsequence has been filtered along the motion trajectory prior to being temporally subsampled by 16. In addition to the $t - L4$ sequence, we must send the $t - H$ subbands at levels 1 through 4, for a total of 16 frames per block. Of course, it is also necessary to transmit all the motion vectors d .

One issue with the MCTF approach is delay. A four-stage MCTF decomposition requires 16 frames, which may be too much delay for some applications, such as visual conferencing. However, one can combine the nonrecursive MCTF with the hybrid coder by limiting the number of MCTF stages and then applying *displaced frame difference* (DFD) prediction (i.e., a hybrid coder such as MPEGx) to the $t - Lk$ subband. In a video conferencing scenario, one may want to limit this kind of built-in *algorithmic delay* to 100 msec, which implies a frame rate greater than or equal to 10 fps. At this frame rate, no stages of MCTF could be allowed. If we raise the frame rate to 20 fps, then we can have one stage. At 50 fps, we can have two stages, etc., and still meet the overall goal of 100-msec algorithmic delay. So as the frame rate goes up, the number of MCTF stages can grow too, even in the visual conferencing scenario. Of course, in streaming video, there is no problem with these small amounts of delay, and one can use five and six stages of MCTF with no difficulty due to algorithmic delay.

The Choi and Woods coder [25] had two levels of MCTF with two-tap Haar filters and spatial wavelet analysis with Daubechies D4 filters and variable block-size motion compensation. This was followed by UTQs with adaptive arithmetic coding and prediction based on subband finite-state scalar quantization (FSSQ) coding technique [26]. An optimized rate allocation was performed across 3-D subbands using the Lagrangian method. The resulting coding performance in terms of PSNR at 1.2 Mbps is shown in Table 12.4–1 for various CIF test clips.

Table 12.4–1 Choi and Woods' Coder Results at 1.2 Mbps

Sequence	$Y(\text{dB})$	$U(\text{dB})$	$V(\text{dB})$
<i>Mobile</i>	27.0	31.2	31.4
<i>Table Tennis</i>	33.8	39.4	39.8
<i>Flower Garden</i>	27.6	32.6	30.8

12.5 SCALABLE VIDEO CODERS

In many applications of video compression, it is not known to the encoder what resolutions, frame rates, and/or qualities (bitrates) are needed at the receivers. Further, in a streaming or broadcast application, different values of these key video parameters may be needed across the communication paths or links. Scalable coders [27] have been advanced to solve this problem without the need for complicated and lossy transcoding.

Definition 12.5–1: Scalability

One coded video bitstream that can be efficiently broken up for use at many spatial resolutions (image sizes), frame rates, regions of interest, and bitrates.

A scalable bitstream that has all four types of scalability will be referred to as *fully scalable*. We can think of region of interest capability as a restricted type of object-based scalability that can, together with resolution scalability, support zoom and pan functionality. This capability can support browsing, wherein a small version of a high-resolution image may be “zoomed and panned” to locate interesting parts for closer looks at higher resolution.

Scalability is needed in several areas, including digital television, heterogeneous computer networks, database browsing, matching of various display formats (to adjust to window size on screen), matching of various display frame rates, loading of a VOD server, etc. Scalability in database browsing facilitates efficient pyramid search of image and video databases. Scalability on heterogeneous networks can enable a network manager to do dynamic load control to match link capacities as well as terminal computer capabilities. A scalable encoder can also better match a variable *carrier-to-noise ratio* (CNR) on wireless channels.

The early standard coders H.263 and MPEG 2 have a limited amount of scalability, usually just one or two levels. For spatial scalability, MPEG 2 uses a pyramid coding method, where the base layer is coded conventionally, and then an enhancement layer supplements this base layer for higher resolution, frame rate, or quality. [Figure 12.5–1](#) shows an MPEG 2 coder targeted for spatial scalability in HD and SD television.

Separate spatial and temporal scalability profiles, with two or three levels only, were standardized in MPEG 2, but are seldom used in practice. The limitations of these scalable profiles are lack of data conservation, limited range of scalability, coding errors not limited to the baseband, and drift for the frequency scalable coder (up to 7 dB over 12 frames [28]).

In the research area, Burt and Adelson [29] introduced a scalable pyramid coder with the desirable quantizer-in-loop property that can frequency shape (roll-off) lower resolution layers. Unfortunately, due to use of Gaussian-Laplacian lowpass pyramid, there is lack of data conservation. Also due to the pyramid based coding-and-recoding

structure, the coding noise and artifacts spread from low (baseband) to higher spatial frequencies. Naveen’s multiresolution SWT coder [19, 30], as sketched in Figure 12.4–3, can be scalable in resolution with three spatial levels. There is no drift at lower resolution because the same motion information is used at the encoder as at the decoder. It uses efficient hierarchical MC that can incorporate rate constraints and frequency roll-off to make the lower resolution frames more videolike (i.e., reduce their spatial high frequencies). Table 12.5–1 gives some average luma PSNR values for HD test clips MIT and *Surfside*, obtained using forward MC in this coder.

The Taubman and Zakhor [31] MCTF algorithm featured global pan-vector motion compensation, layered (embedded) quantization, and conditional zero coding to facilitate SNR scalability. The published PSNR results for two common CIF test clips at 1.5 Mbps, coding only the luma (*Y*) component are shown in Table 12.5–2. Results were good on panning motion clips but less efficient on clips with detailed local motion. Also provided is an MPEG 1 result for comparison.

Woods and Lilienfield [32] published results on a *resolution scalable coder* (RSC), which performed spatial subband analysis first, and then one stage of MCTF at each resolution. RSC then applied the hybrid technique of DFD on the lowest spatiotemporal level to yield three spatial resolutions and two frame rates. The approach was nonembedded in bitstream and used UTQs based on the generalized Gaussian

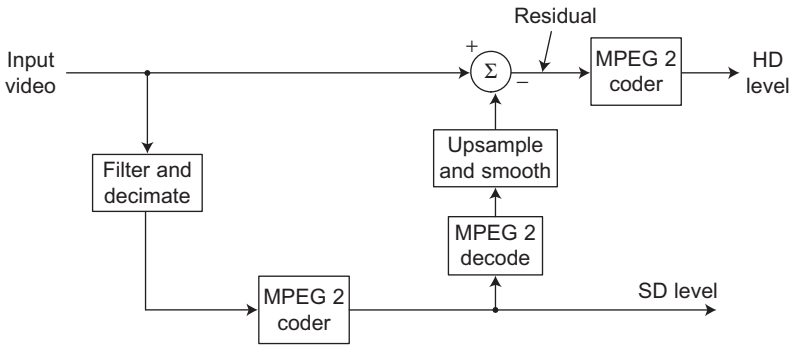


FIGURE 12.5–1

Illustration of resolution scalability for SD and HD video using MPEG 2.

Table 12.5–1 PSNR Results—Forward Motion Compensation		
PSNR (dB)	MIT (0.5 bpp)	Surfside (0.25 bpp)
Low res.	38.0	42.2
Medium	33.5	41.6
High res.	34.3	42.5
High (not scal.)	35.0	42.8

distribution together with multistage quantizers to achieve both resolution and frame-rate scalability. The RSC coder is of the type “2-D+t” since it first performs a spatial subband/wavelet pyramid on the incoming data in Figure 12.5–2. The various spatial levels are then fed into stages of motion estimation and MCTF, with outputs shown on the right, consisting of temporal low L_i and high H_i frames. Referring to Figure 12.5–2, the low frame rate video hierarchy is given as V_{i0} , and the high rate hierarchy, denoted V_{i1} , is obtained by the addition of the corresponding H_i frame. Average PSNR results for the luma channel are given in Table 12.5–3, where each channel or spatiotemporal resolution was coded at 0.35 bpp. We see that the high-frequency roll-off is moderately successful at keeping the PSNR relatively constant across the levels. Without such roll-off, the low resolution frames appear excessively sharp and may contain alias error.

Hsiang and Woods [33] presented an MCTF scalable video coder, wherein an invertible motion-compensated 3-D SWT filter bank was utilized for video analysis/synthesis. The efficient embedded zero-block image coding scheme EZBC [34] was extended for use in coding the video subbands. Comparison on *Mobile* in CIF format to a commonly available MPEG 2 coder showed PSNR improvements ranging from 1 to 3 dB over a broad range of bitrates for this scalable coder.

Table 12.5–2 Motion Compensation via Global Pan Vector at 1.5 Mbps

PSNR (dB)	MPEG 1	Highly Scalable
<i>Football</i>	33.6	34.6
<i>Table Tennis</i>	32.8	34.6

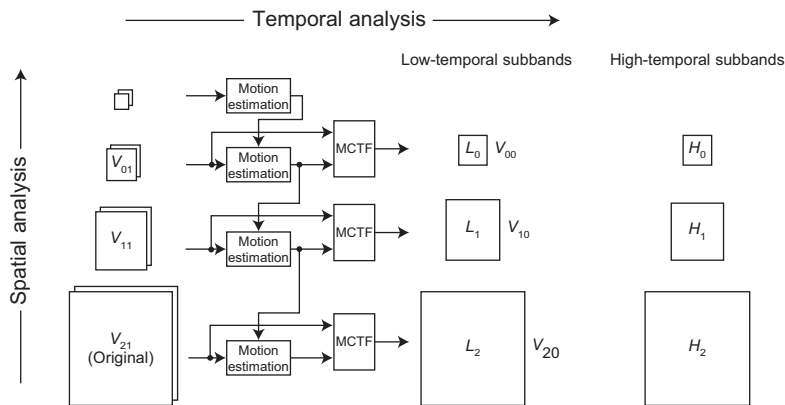


FIGURE 12.5–2

RSC demonstration software: three resolutions and two frame rates (© IEEE).

Table 12.5–3 RSC Average PSNR (dB) on HD Test Clips Mall and MIT

PSNR (dB) video level		Low frame rate			High frame rate		
		V_{00}	V_{10}	V_{20}	V_{01}	V_{11}	V_{21}
Mall	w/o roll-off	33.8	37.3	42.3	34.5	37.0	40.7
	with roll-off	36.0	39.4	42.2	37.2	39.5	40.6
MIT	w/o roll-off	30.4	28.7	32.1	31.9	31.1	33.2
	with roll-off	33.1	31.7	32.0	34.8	34.1	33.2

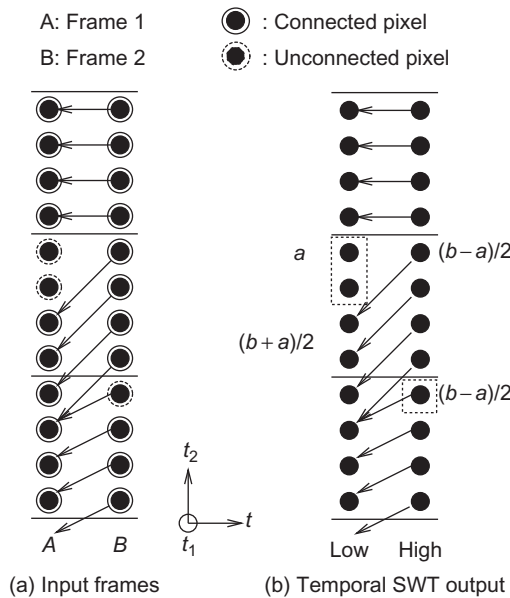


FIGURE 12.5–3

Illustration of MCTF using Haar filters.

More on MCTF

MCTF plays a very important role in interframe SWT coding. In general, we try to concentrate the energy onto the temporal low subband, with the energy in the temporal high subbands thus as small as possible. But *covered* and *uncovered* pixels must be considered in achieving this goal. With reference to the Haar filter MCTF shown in Figure 12.5–3(a), let frame A be the reference frame; then some pixels in frame B will be uncovered pixels, meaning they will have no reference in frame A. In Figure 12.5–3(b) we show the resulting $t-L$ and $t-H$ frames, as computed in place, with the horizontal lines indicating a smallest block-matching size of 4×4 . We see many connected pixels, but some in frame A are unconnected. They are either covered in frame B or else are just not the best match. The latter situation can happen

due to expansion and contraction of motion fields as objects move toward or away from the camera.

We can see pixels in Figure 12.5–3 classified as *connected* and *unconnected*. If there is a one-to-one connection between two pixels, they are classified as connected pixels. If several pixels in frame *A* connect to the same pixel in frame *B*, only one of them is classified as the connected pixel; the others are declared unconnected. These unconnected pixels in frame *B* are indicated by no reference between frame *A* and *B*. After the classification, we can perform the actual MC filtering on the connected pixels. For the unconnected pixels in frame *B*, their scaled original values are inserted into the temporal low subband without creating any problem. For the unconnected pixels in frame *A*, a scaled DFD can be substituted into the temporal high subband [35], if there is a pixel in frame *B* or some earlier or later frame pair⁴ that has a good (but maybe not best) match. Otherwise, some spatial or intraframe prediction/interpolation can be used, analogous to an *I* block in MPEG-coded video.

Detection of Covered Pixels

In order to make the covered pixels correspond to the real occurrence of the occlusion effect, we can first use a detection phase before the actual MC filtering. Here is a simple pixel-based algorithm to find the true covered pixels illustrated for the Haar MCTF case.

Covered Pixel Detection Algorithm

There are four steps, as illustrated in Figure 12.5–4, for the Haar MCTF case:

Step 1. Do backward motion estimation from frame *B* to frame *A*, in a frame pair.

Step 2. Get the state of connection of each pixel in frame *A*. There are three states:

Unreferred: a pixel that is not used as reference.

Uni-connected: a pixel that is used as reference by only one pixel in frame *B*.

Multi-connected: a pixel that is used as reference by more than one pixel in the current frame.

A multi-connected pixel in frame *A* has several corresponding pixels in frame *B*, so we compute the absolute DFD value with each of them and just keep the minimum as uni-connected.

Step 3. Get the state of connection of each pixel in frame *B*. There are just two states:

Uni-connected: Here we have three cases:

Case 1: a pixel whose reference in frame *A* is uni-connected.

Case 2: a pixel whose reference in frame *A* is multi-connected; except if its absolute DFD value with the reference pixel is the only minimum, we declare it uni-connected.

⁴Remember, this is all feed-forward, so any future reference only incurs a small additional delay at the transmitter and receiver. Of course, this could be a problem in some applications, such as visual conferencing.

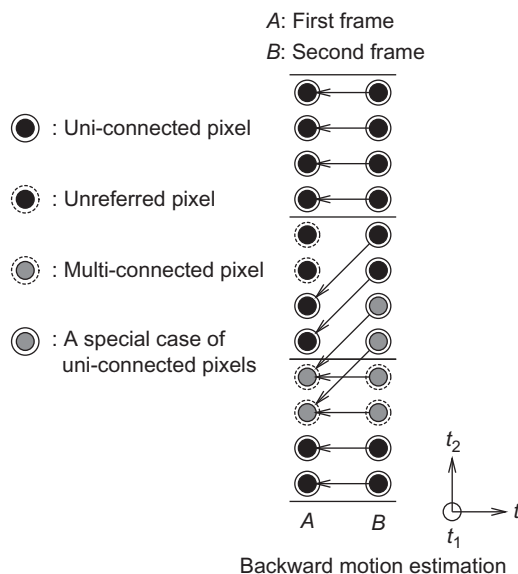


FIGURE 12.5-4

Illustration of the covered pixels.

Case 3: if there are several pixels in frame *B* pointing to the same reference pixel, and having the same minimum absolute DFD value, we settle this tie using the scan order.

Multi-connected: remaining pixels in frame *B*.

Step 4. If more than half of the pixels in a 4×4 block of frame *B* are multi-connected, we try forward motion estimation. If motion estimation in this direction has smaller MCP error, we call this block a covered block and pixels in this block are said to be covered pixels. We then use forward MCP to code this block. We thus have created a bidirectional motion field.

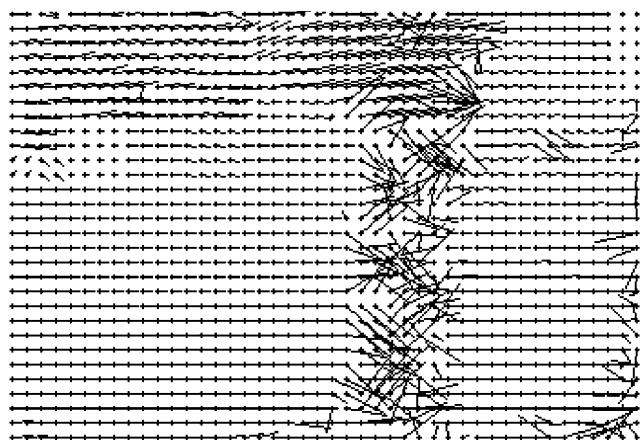
In Figure 12.5-5, we see the detections by this algorithm of the covered and uncovered regions, mainly on the side of the tree, but also at some image boundaries. With reference to Figure 12.5-6, we see quite strange motion vectors obtained around the sides of the tree as it “moves” due to the camera panning motion in *flower garden*. The bidirectional motion field is quite different, as seen in Figure 12.5-7, where the motion vector arrows generally look much more like the real panning motion in this scene. These figures were obtained from the MCTF operating two temporal levels down—that is, with a frame separation of four times that which is normal.

Bidirectional MCTF

Using the covered pixel detection algorithm, the detected covered and uncovered pixels are more consistent with the true occurrence of the occlusion effect. We can

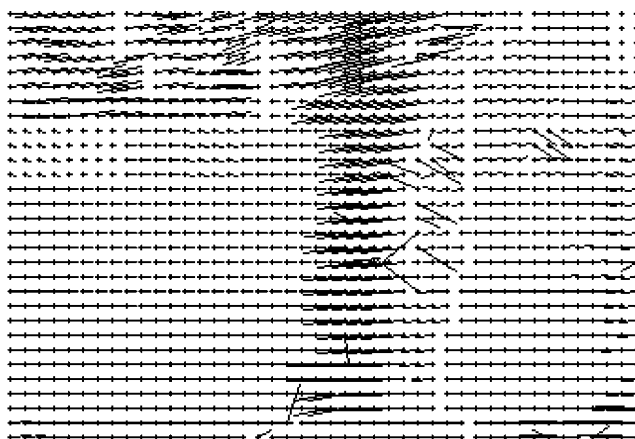
**FIGURE 12.5-5**

Example of unconnected blocks detected in *flower garden*.

**FIGURE 12.5-6**

Example of unidirectional motion field.

now describe a bidirectional MCTF process. We first get backward motion vectors for all 4×4 blocks via HVSBM. The motion vectors typically have quarter- or eighth-pixel accuracy, and we use the MAE block-matching criterion. Then we find covered blocks in frame *A* using the covered pixel detection algorithm. The forward motion estimation is a by-product of the process of finding the covered blocks. Then we

**FIGURE 12.5-7**

Example of bidirectional motion field.

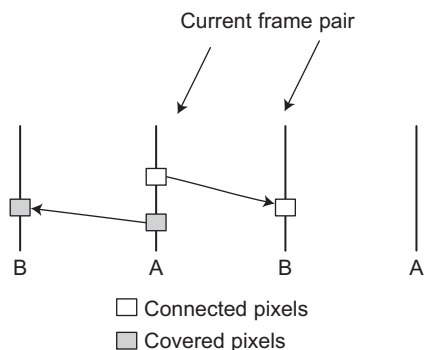
**FIGURE 12.5-8**

Illustration of bidirectional interpolation in the context of two-tap Haar MCTF.

do optimal tree pruning [35], and thus we realize a bidirectional variable block size motion compensation.

Figure 12.5-8 shows a variation on the bidirectional MCTF, where we first choose forward motion compensation, and then resort to backward motion compensation only for detected covered blocks. The shaded block in frame A is covered. For those covered pixels, we perform MCP. For the other regions, the MCTF is similar to that in [35].

In either case, at GOP boundaries, we have to decide whether to use open or closed GOP, depending on delay and complexity issues.

**FIGURE 12.5-9**

A frame output from unidirectional MCTF at four temporal levels down.

**FIGURE 12.5-10**

Output of bidirectional MCTF at four temporal levels down.

We show in [Figure 12.5-9](#) a $t - L4$ output from a unidirectional MCTF at temporal level four (i.e., one-sixteenth of the original frame rate). There are a lot of artifacts evident, especially in the tree. [Figure 12.5-10](#) shows the corresponding $t - L4$ output

from a bidirectional MCTF; we can detect a small amount of blurring but no large artifact areas as produced by the unidirectional MCTF.

Some MC-EZBC experimental coding results using the bidirectional Haar MCTF are contained in Chen and Woods [36] for the CIF test clips mobile, flower garden, and *coast guard*. Some MC-EZBC results in coding of 2K digital cinema data are contained in the MCTF video coding review article [37]. Video comparisons of MC-EZBC to both H.264/AVC and MPEG 2 results are available at this book's Web site.

Enhanced MC-EZBC

An enhanced version of MC-EZBC appeared in Wu et al. [38] that combined an adaptive LGT/Haar MCTF [39] with scalable motion vector coding and OBMC. The adaptive MCTF is shown in Figure 12.5–11, borrowed from [38]. It shows across the top an incoming video, broken into two frame phases, denoted A and B. The more efficient 5/3 LGT filter is preferred, but the two-tap Haar filter is used at scene change and when the motion compensation error is large. We can see the adaptive 5/3 MCTF as a generalization of the bidirectional Haar MCTF covered above.

A Lagrange-based comparison is made of eight different coding modes, weighting both MC error and MV rate on a block-by-block basis. A coding result from this

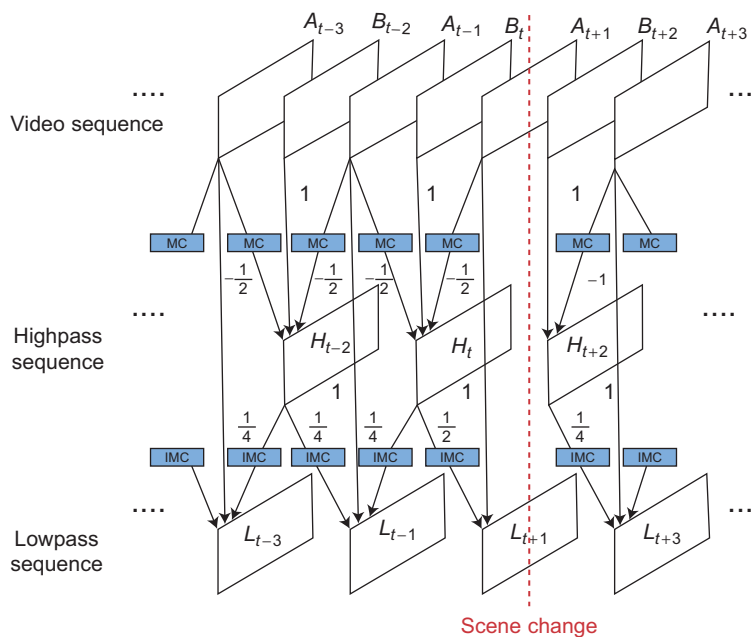
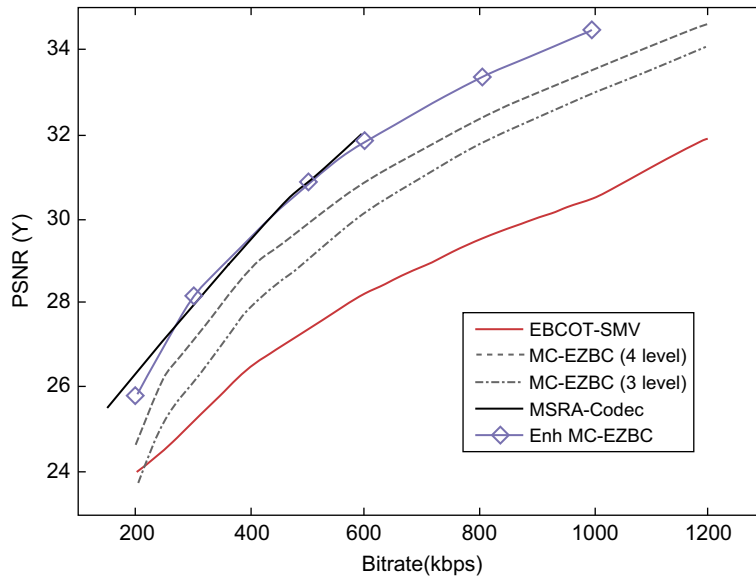


FIGURE 12.5–11

Adaptive LGT/Haar MCTF. (Figure 1 from [38] © IEEE 2008)

**FIGURE 12.5-12**

A coding comparison.

work is shown in Figure 12.5-12 for the PSNR(Y) of the test clip *Bus* at full (CIF) resolution and full (30 fps) frame rate. Two curves are plotted; one for four temporal levels of MCTF and a lower performance curve for three temporal levels. Also plotted for comparison purposes are two other transform coders. The one labeled MSRA-Codec is taken from [40], and the EBCOT-SMV result with three temporal levels is taken from [41]. Also shown in [38] is a comparison with an early version of H.264/AVC that shows a rough parity of performance in terms of luminance PSNR.

12.6 CURRENT INTERFRAME CODING STANDARDS

In this section we discuss the modern coding standards developed jointly by MPEG and the Video Coding Experts Group (VCEG), collectively denoted as H.264/xxx. The first one, H.264/AVC, is for nonscalable coding of video and achieves a 50% bitrate savings over MPEG 2. The second one, H.264/SVC, is for scalable video coding and claims a 10% penalty on average for providing scalability. The last, H.264/MVC, is for stereo and other multiview applications, where MVC stands for multiview coding; it has been released most recently and is an area still currently under development by these standards groups.

H.264/AVC

Research on increasing coding efficiency continued through the late 1990s, and it was found that up to a 50% increase in coding efficiency could be obtained by various improvements to the basic hybrid coding approach of MPEG 2. Instead of using one hypothesis for the motion estimation, multiple hypotheses from multiple locations in previous frames could be used [42] together with a Lagrangian approach to allocate the bits. By 2001 the video standards groups VCEG at ITU, and MPEG ISO convened a joint video team (JVT) to work on the new standard, called H.264 by ITU/VCEG and Advanced Video Coder (AVC) by the ISO/MPEG. The resulting H.264/AVC coder is also sometimes referred to as MPEG 4, Part 10 and is illustrated in Figure 12.6–1, where we see that more frame memory to store past frames has been added to the basic hybrid coder of Figure 12.2–2. The H.264/AVC coder is free to choose any of the frames in its multi-frame memory, termed the decoded picture buffer (DPB), to perform the prediction. We also see the addition of a loop filter that serves to smooth out blocking artifacts. Further, before the intra transform, there is intra- or directional spatial prediction that improves coding efficiency, but introduces the need to switch between these new intra modes, as well as the interprediction modes.

There are many new ideas in H.264/AVC, which allow it to perform at twice the efficiency of the MPEG 2 standard, also known as H.262. There is a new variable block size motion estimation with block sizes ranging from 16×16 down to 4×4 , and motion vector accuracy raised to quarter pixel from the half-pixel accuracy of MPEG 2. The permitted block size choices are shown in Figure 12.6–2. The 16×16

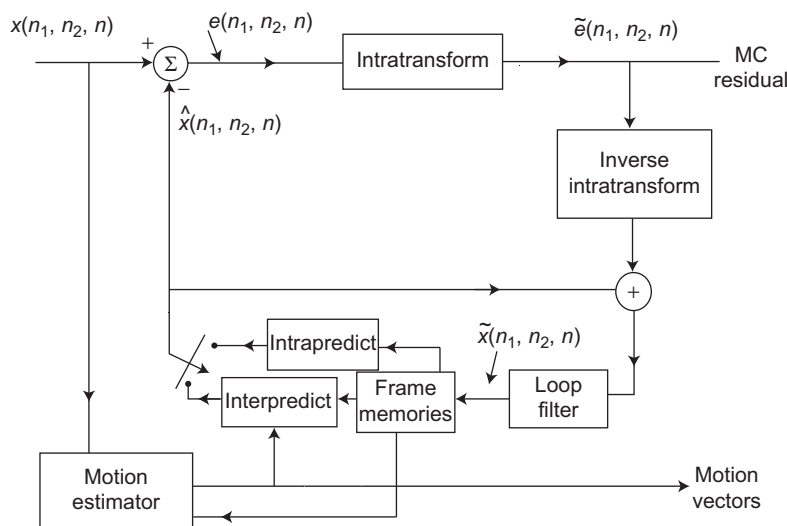
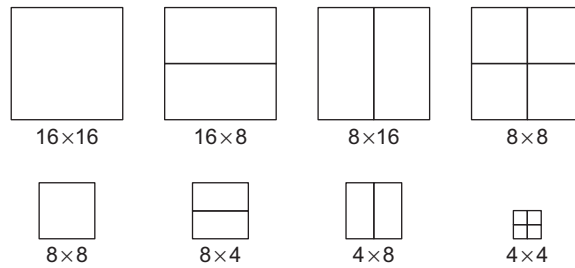


FIGURE 12.6–1

A system diagram of the H.264/AVC coder.

**FIGURE 12.6-2**

Allowed MV block sizes in H.264/AVC.

macroblock can be split in three ways to get 16×8 , 8×16 , or 8×8 , where by 16×8 splitting we mean split into two 16×8 submacroblocks, as shown. If the prediction accuracy of 8×8 blocks is not enough, one more round of such splitting finally results in the submacroblocks 8×4 , 4×8 , or 4×4 . Note that in addition to what we would get by quadtree splitting (see Chapter 11), we get the possible rectangular blocks, which can be thought of as a level inserted between two quadtree block levels.

To match this smallest block size, a 4×4 integer-based transform that is DCT-like is introduced, and separable based on the 1-D four-point transform

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \quad (12.6-1)$$

The H.264/AVC coder is based on *slices*, with *I*, *B*, and *P* slices, as well as two new *switching slices* *SP* and *SI*. The slices are in turn made up of 16×16 macroblocks. The *P* slice can have *I* or *P* macroblocks. The *B* slice can have *I*, *B*, or *P* macroblocks. There is no mention of GOP, but there are *I* pictures, needed at the start to initialize this hybrid coder. Of course, there is nothing to prohibit a slice from being the size of a whole frame, so that there can effectively be *P* and *B* frames as well.

In H.264/AVC, an *I* slice is defined as one whose macroblocks are all intracoded. A *P* slice has macroblocks that can be predictively coded with up to *one* motion vector per block. A *B* slice has macroblocks predictively (interpolatively) coded using up to *two* motion vectors per block. Additionally, there are new switching slices *SP* and *SI* that permit efficient jumping from place to place within or across bitstreams (cf. Section 13.3).

Within an *I* slice, there is *intrapicture prediction*, done blockwise based on previously coded blocks. The intraprediction block error is then subject to the 4×4 integer-based transform and quantization. The intraprediction is adaptive and directional based for 4×4 blocks, as indicated in Figure 12.6-3. A prespecified fixed blending of the available boundary values is tried in each of the eight prediction directions. Also available is a so-called DC option that predicts the whole block as a

constant. For 16×16 blocks, only four intraprediction modes are possible: vertical, horizontal, DC, and *plane*, the last one coding the values of a best-fit plane for the macroblock. The motion compensation can use multiple references, so for example, a block in a *P* slice can be predicted by one to four reference blocks in earlier frames. The standard specifies the amount of reference frame storage in the DPB, that must be available to store these past pictures.

Figure 12.6–4 illustrates the comparative PSNR versus bitrate performance of the test models of some of the MPEG/VCEG coders on the 15 fps CIF test clip *Tempete*. The figure is from Sullivan and Wiegand’s review article [43] and shows considerable improvement versus MPEG 2, of about a factor of two increased compression

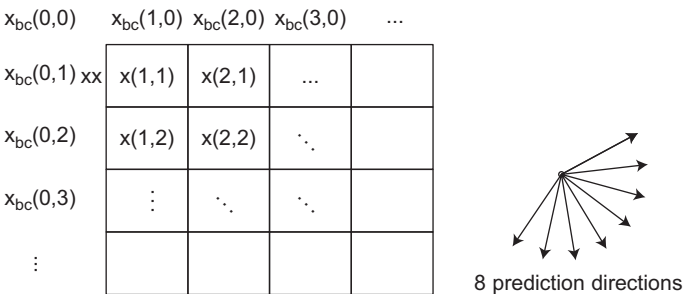


FIGURE 12.6–3

Illustration of directional prediction modes of H.264/AVC in the case of 4×4 blocks.

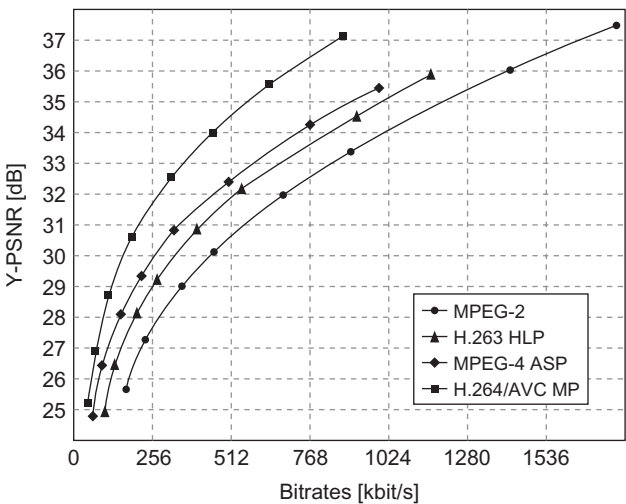


FIGURE 12.6–4

PSNR vs. bitrate for the 15 fps CIF test clip *Tempete*. (© IEEE 2005.)

at fixed PSNR. This improvement in efficiency is largely due to the greater exploitation of motion information made possible by the revolutionary increases in affordable computational power of the last 10 years. More information on the H.264/AVC standard is available in the review article by Wiegand et al. [44] that introduces a special journal issue on this topic [45].

Video Coder Mode Control

A video coder like H.264/AVC has many coder modes to be determined. For each macroblock, there is the decision of inter or intra, quantizer step size, and motion vector and block modes. Each choice has a different number of coded bits for the macroblock. We can write the relevant Lagrangian form for block k as

$$J_k \triangleq D_k(\mathbf{b}_k, \mathbf{m}_k, Q_s) + \lambda_{\text{mode}} R_k(\mathbf{b}_k, \mathbf{m}_k, Q_s), \quad (12.6-2)$$

where D_k and R_k are the corresponding mean-square distortion and rate for block \mathbf{b}_k when coding with mode \mathbf{m}_k and Q_s is the quantizer step size. Here, R_k must include the quantizer bits for coding the block plus estimated bits for the motion vector(s) and mode decisions. For the moment, assume that the motion vectors have been determined prior to this optimization. Then we can sum over all the blocks in a frame to get the total distortion and rate for that frame

$$\begin{aligned} J &= D + \lambda_{\text{mode}} R \\ &= \sum_k \{D_k(\mathbf{b}_k, \mathbf{m}_k, Q_s) + \lambda_{\text{mode}} R_k(\mathbf{b}_k, \mathbf{m}_k, Q_s)\} \\ &= \sum_k J_k, \end{aligned}$$

where D and R are the total distortion and rate. Normally the blocks in a frame are scanned in the 2-D raster manner (i.e., left to right and then top to bottom). The joint optimization of all the modes and Q_s values for these blocks is a daunting task. In practical coders, often what optimization there is, is done macroblock by macroblock, wherein the best choice of mode \mathbf{m}_k and Q_s is done for block \mathbf{b}_k conditional on the past of the frame in the NSHP sense, resulting in at most a stepwise optimal result. We can achieve this stepwise or greedy optimal result by evaluating J_k in (12.6-2) for all the modes \mathbf{m}_k and quantizer step sizes Q_s and then choosing the lowest value. This point will then be on the optimized operational $D(R)$ curve for some rate R_k . To generate the entire curve, we would then sweep through the parameter λ_{mode} . This is still too much computation for a practical coder, so in an exhaustive set of experiments, Wiegand and Girod [46] searched through λ_{mode} values for a wide range of Q values and came up with the following equation that is best in an average sense:

$$\lambda_{\text{mode}} = cQ_s^2,$$

with the value $c = 0.85$. This work, first done on a test model for H.263, then becomes: Choose the mode m_k such that

$$m_k = \arg \min_k \left\{ D_k(\mathbf{b}_k, \mathbf{m}_k, Q_s) + cQ_s^2 R_k(\mathbf{b}_k, \mathbf{m}_k, Q_s) \right\}.$$

In the case of inter mode, the motion vector must also be determined, which is done using a similar Lagrangian form with a measure of DFD error in place of coding distortion and also motion vector rate in place of total rate [42, 46]. If MSE is used as the distortion then the same lambda value is recommended (i.e., $\lambda_{\text{motion}} = \lambda_{\text{mode}}$), but if MAE is used, then $\lambda_{\text{motion}} = \sqrt{\lambda_{\text{mode}}}$ is used.

When these experiments were redone for the H.264/AVC test model, with its larger set of coding and motion modes, a similar formula emerged. However, since the quantizer parameter Q_p was now defined logarithmically in the step size, the formula that emerged [47] was

$$\lambda_{\text{mode}} = c2^{(Q_p-12)/3}.$$

In both cases, this operational *rate-distortion optimization* at constant Q_p , results in a sequence of macroblocks approximately optimized in the so-called constant-slope sense. To actually get a CBR-coded sequence, a further optimization (called simply *rate control* in H.264/AVC) has to be applied to decide what value of Q_p to use for each macroblock. An easy VBR case results from the choice of constant Q_p , but then the total bitrate is unconstrained and a search over Q_p must be done to get the target bitrate at the end of the frame, GOP, or movie. Also in practice, different Q_p values are used for *I*, *P*, and *B* slices or frames. Often this choice is fixed, with step size increasing in a predetermined manner (usually +1 or +2) as we move from *I* to *P* to *B* frames. Recent frame-by-frame optimization results for H.264/AVC have been reported in [48], which show a 6–12% decrease in bitrate on QCIF sequences versus the test model software JM8.2 of the joint video team [49].

Network Adaptation

In H.264/AVC, there is a separation of the *video coding layer* (VCL), which we have been discussing, from the actual transport bitstream (TS). As in MPEG 4 video, H.264/AVC is intended to be carried on many different networks with different packet or cell sizes and qualities of service. So MPEG simply provides a *network abstraction layer* (NAL) specification about how to adapt the output of the VCL to any of several transport streams. There is much more on network video in Chapter 13, where we further discuss some basic issues in network video. More on H.264/AVC is found in Chapter 10 of *The Essential Guide to Video Processing* [50].

H.264/SVC

In 2007 the ISO and ITU, via their MPEG and VCEG committees, issued an amendment to H.264/AVC for scalable coding. The new Scalable Video Coder (SVC) is a

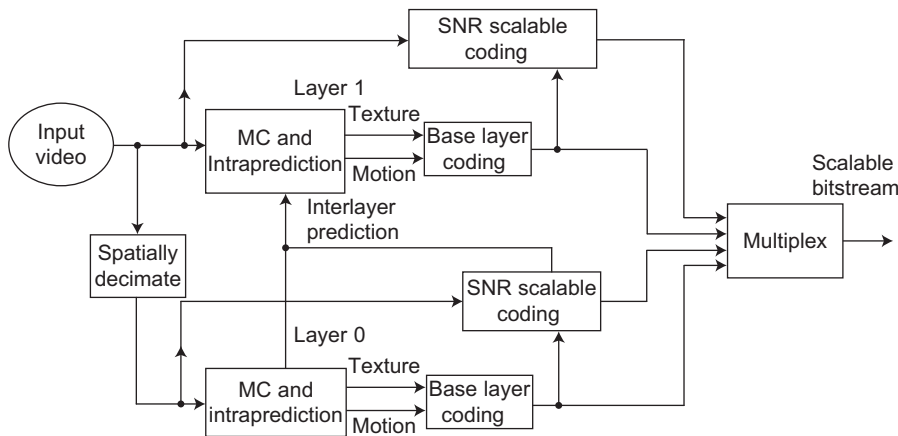


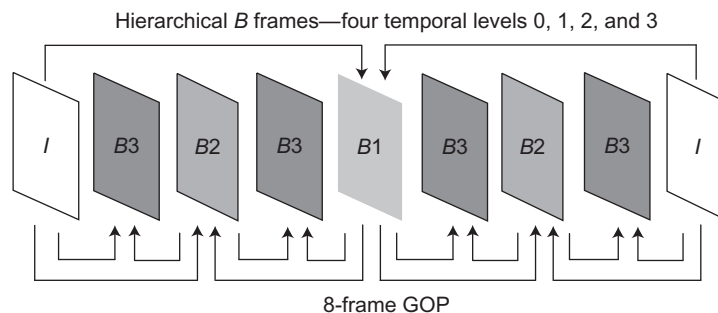
FIGURE 12.6-5

MPEG Scalable Video Coder.

layered scalable coder that has an H.264/AVC base layer and uses many of the AVC tools in the enhancement layers [51]. A simplified diagram in Figure 12.6-5 shows a two-layer *spatial scalability* version of SVC. Layer 0 is a conventional H.264/AVC coder with the addition of SNR or quality scalability of its *texture* (residual) output and the provision of so-called *interlayer prediction* quantities to help the Layer 1 (enhancement layer) coder.

Comparing to the modes possible in AVC, here we have additional modes for possible prediction from the present value of the reference layer, Layer 0 in Figure 12.6-5. There are two interlayer prediction modes: *interlayer texture prediction*, available only when the reference macroblock is completely intramode, and *interlayer residual prediction*, only available when the reference block is completely in intermode. Since the quantized version of this Layer 0 residual can be decoded directly at the receiver, it is not necessary to run a separate motion compensation loop when decoding Layer 1. Both interlayer predictions are upsampled with H.264/SVC specified filters. The interlayer texture prediction is directly used as the Layer 1 predictor, but the inter-layer residual prediction is added to the Layer 1 motion-compensated prediction to form the Layer 1 prediction.

Another key concept in H.264/SVC is the use of *hierarchical B frames*, as illustrated in Figure 12.6-6 showing a *temporal scalability* version of SVC. We see the example with a GOP of length 8 starting and ending on an I frame, the I frame subsequence thus at 8 times lower frame rate, termed temporal level T_0 . Two neighboring I frames are then used to predict (interpolate) a B frame in the middle marked $B1$, thus creating a sequence at four times lower than the full frame rate, called temporal level T_1 . Then $B2$ frames are created by interpolation at two frames distance, creating the T_2 temporal level. Continuing this procedure one more time, we get the full frame rate, T_3 temporal level. We now have a hierarchy of B frames with coding

**FIGURE 12.6-6**

Hierarchical *B* frame concept.

order indicated by their numbers. Interestingly, this hierarchy of *B* frames is identical to what happens in a 5/3 MCTF if we omit the update terms [37]. Such hierarchical interpolation structures have been found to be very efficient and are consistent with the general reference frame structure of H.264/AVC where they can offer more than 1-dB advantage over IBBP... type structures [52] for CIF-sized test clips at 30 fps and GOP size of 16. The drawback, though, is a *structural or algorithmic delay* that varies within the GOP, and in the case of Figure 12.6-6 is seven frames for the second picture (frame) in the GOP. Lower structural delays are possible by limiting the use of forward prediction, but at a penalty in performance.

The use of an *I* frame every 8 frames is rather inefficient but permits high random access. A more efficient structure would combine hierarchical *B* frames with *P* frames, as shown in Figure 12.6-7.⁵ The spatial and temporal scalability can be combined to produce a spatiotemporal scalable version of SVC, such as QCIF at 15 fps and CIF at 30 fps. Results are shown for this scenario for the *Football* test clip in Figure 12.6-8 showing the SVC results obtained with the JSVM software v. 9.1 along with single-layer results of AVC and the simulcast solution—that is, the solution when two separate single layers are sent, one for each spatiotemporal resolution. In this figure, just the results for the CIF layer are plotted, since the QCIF base layer is coded by an H.264/AVC compatible coder. Hierarchical *B* frames are used here for all coders, as they had been found to be more efficient. The plotted points here correspond to separate encoder runs and are connected together for plotting purposes only. We see that the SVC result is above the midpoint between single layer and simulcast. In later work, joint optimization was tried [53] to produce an SVC coder more balanced between base and enhancement layers, with results shown in [54]. This review

⁵In connection with these two figures on hierarchical *B* frames, it is important to note that for H.264/AVC style coders, these diagrams do *not* indicate the frames used in prediction since multiple frames can be used from the already-coded frame memory. The figures *do* indicate the highest temporal level of the frames that can be used though—i.e., all already-coded frames lower than the present one.

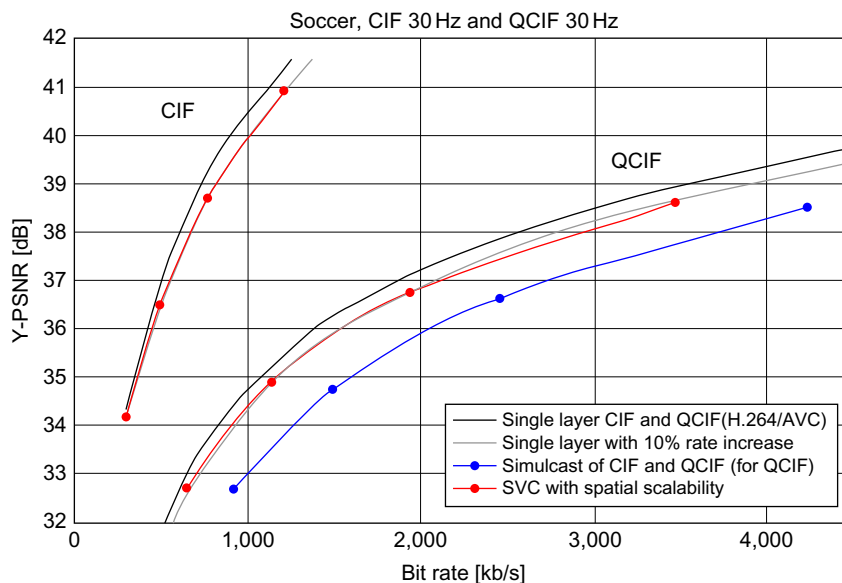
**FIGURE 12.6-9**

Illustration of H.264/SVC performance for spatiotemporal scalability—QCIF at 30 fps and CIF at 15 fps (from [54], © IEEE 2008).

for quality scalability alone was also given in [54] and is shown in Figure 12.6-10 which again shows a 10% penalty in rate, here for 6 layers of quality scalability. All six plotted points come from the same encoded stream, with successively discarding of so-called quality enhancement packets. The various types of scalability may be combined, but results were not shown. It is expected that performance will suffer.

We have seen that the H.264/SVC coders are layered with motion determined from the bottom up for the chosen B -frame temporal hierarchy. For comparison purposes, the MCTF coders commonly determine their motion first at the finest temporal resolution and then scale it down and adjust it for the lower frame rates. Another difference that should be noted is that embedded MCTF scalable coders (e.g., MC-EZBC) provide simultaneous spatial, temporal, and quality scalability with no restriction to a coarse set of quality layers.

SVC for Video Conferencing

In video conferencing, overall delay is an important factor. Generally it is agreed that this delay should be kept below 120–200 msec to enable participants to talk naturally without pausing to wait for the other to finish. Due to this need, the use of B frames is discouraged in this application because they cause additional delay. Yet hierarchical B frames are very effective for compression efficiency. However, a restricted version

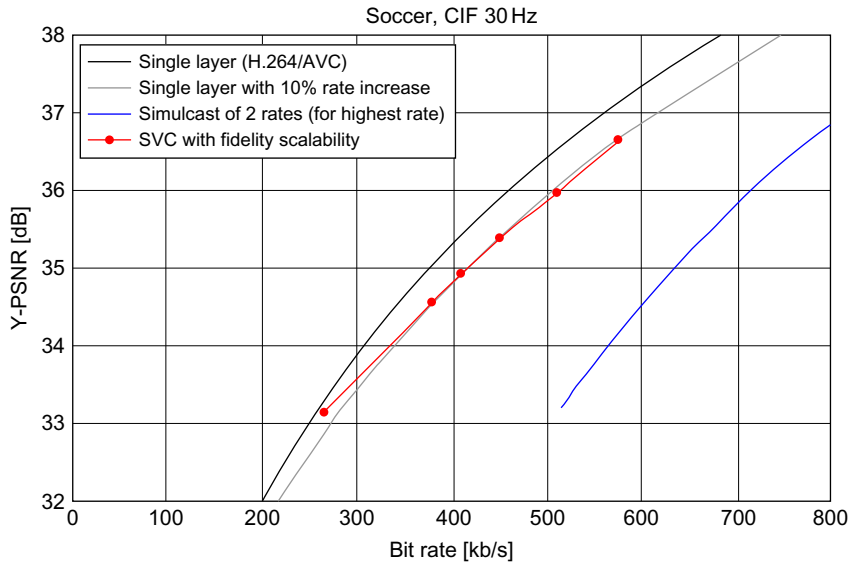


FIGURE 12.6-10

Illustration of quality scalability from the soccer clip with CIF at 30 fps. (from [54], © IEEE 2008)

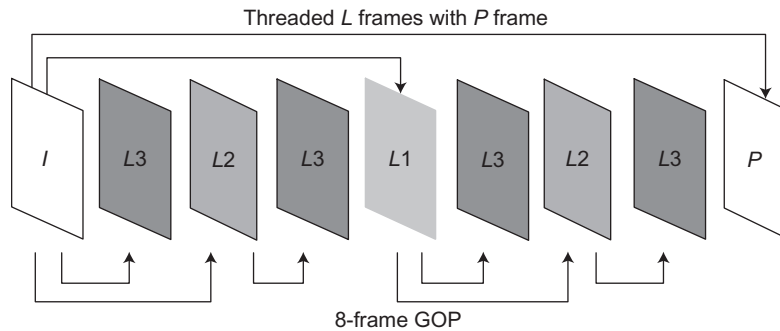


FIGURE 12.6-11

Efficient threaded hierarchical structure without increased structural delay of B frames.

of hierarchical B frames, referred to as *threading* [56], can be used because it only depends on prior coded slices or frames. In Figure 12.6-11, the thread frame type is denoted by L and two levels are shown.

Example 12.6-1: Multipoint Video Conferencing

In video conferencing with multiple participants at separate locations, the link bandwidth and user computational capability often dictate the use of different resolutions, frame

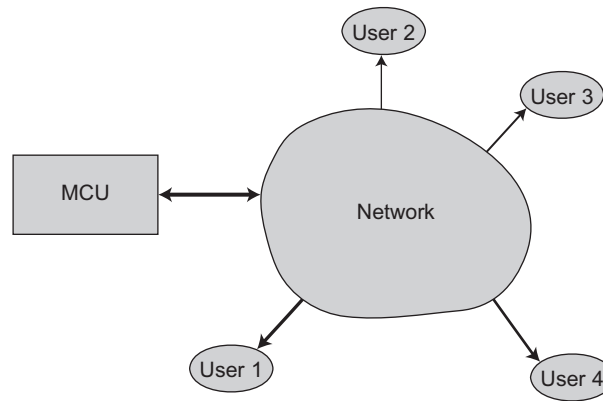
**FIGURE 12.6–12**

Illustration of conventional multipoint video conferencing architecture.

rates, and bitrates for the various receivers. A common method to deal with this problem in the case of nonscalable coders is shown in Figure 12.6–12. This traditional approach [57] uses a device called *multipoint control unit* (MCU) to perform transcoding among the various formats used. We have indicated the various link bandwidths by the thickness of the connecting lines.

Unfortunately, this approach introduces extra delay in the video conference due to the use of transcoding (decode/compose/recode) to serve the various users' requirements. Usually all participants would have to agree on some minimal quality for the video conference, but this is not really satisfactory in many cases. With the advent of SVC, the MCU is not needed because transcoding is not needed. Using the SVC standard, so-called *video routers* can be substituted for the MCUs. These routers only have to forward and delete packets as appropriate to that user's connection and needs, so little extra delay is added. In effect, video packets can be routed by the video router based on their headers, just like other network packets. Each user then has a simple client application that decodes and composes the various videos together into a suitable browser window. Such a system was simulated in [58], and significant advantage was shown in terms of reduced multipoint video conferencing delay. Two channels were employed: one with high reliability (HRC), used for the base layer, and one with low reliability (LRC) and 5% packet loss used for enhancement layers. Figure 12.6–13 from [58], plots luminance PSNR for two systems, one with MCU and one with SVC instead. The curves plotted in the right of the figure are for a single-layer coder and include the MCU delay. The curves plotted on the left incorporate SVC and show improved performance with much smaller delays, due to hierarchical B and threaded L frames. The MCU system is seen to suffer significant extra delay, in the vicinity of 150–200 msec, making it marginal for conversation. The MCU system also suffers more from packet loss, while the SVC system (denoted SVCS for *scalable video conferencing server*) has been able to decode the base layer sent over the HRC and suffered only a

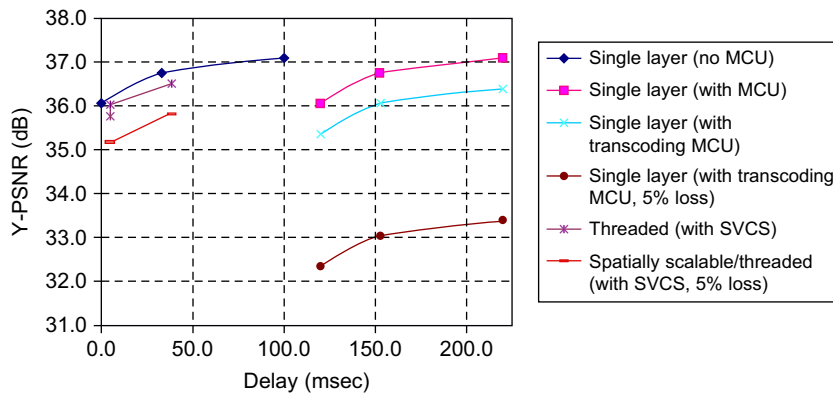


FIGURE 12.6-13

A simulated performance comparison of PSNR versus delay. (from [58] © JZUS A 2006)

small PSNR loss. The H.264/AVC coder has also suffered a partial loss of its bitstream on the LRC, but with much larger consequences due to its non-scalable structure. ■

H.264/MVC

An extension of the H.264/AVC standard for multiview or stereo 3-D coding was released by ITU/ISO in 2008 to support the simultaneous compression of video from several cameras with largely overlapping fields of view. The new standard H.264/MVC uses a hierarchical *B*-frame structure with the addition of inter-view prediction modes, as seen in Figure 12.6-14. In this figure, with five cameras, we first notice the hierarchical *B*-frame temporal hierarchies indicated for coding the individual camera streams. But there are additional coding dependencies shown between the camera (views) representing the inter-view predictions. We see that the first frame in the cam 3 sequence is predicted by the first coded frame in the cam 1 sequence, and then in turn the first frame in the cam 5 sequence. These first frames are in turn used to predict the *B1*-level frames in their individual sequences, and so on in sequences 3 and 5. After the GOPs are complete, the cam 2 and cam 4 sequences are coded with the new option for inter-view prediction from the already-coded frames. Just as we talk of temporal level in the *B*-frame hierarchy, we can talk of *view level* here, meaning the frame's level or coding order, in the inter-view *B*-frame hierarchy.

We see in Figure 12.6-14 that the first *B* frame in the cam 2 sequence can be predicted by the corresponding *B* frames in the cam 1 and cam 3 sequences or by the *B* frame on either side in its own sequence, or by any already-coded frame of lower temporal or view level in the DPB. Again, the diagram does not indicate what frame is used, just what frames cannot be used (i.e., those whose level is higher in terms of inter-view levels).

It was noticed that due to the great flexibility of picture memory referencing allowed in H.264/AVC, the new MVC coding could be conducted with this software by stringing the multiview frames together into a single stream [59]. The DPB must be very large to accommodate this. The objective (PSNR) coding results of

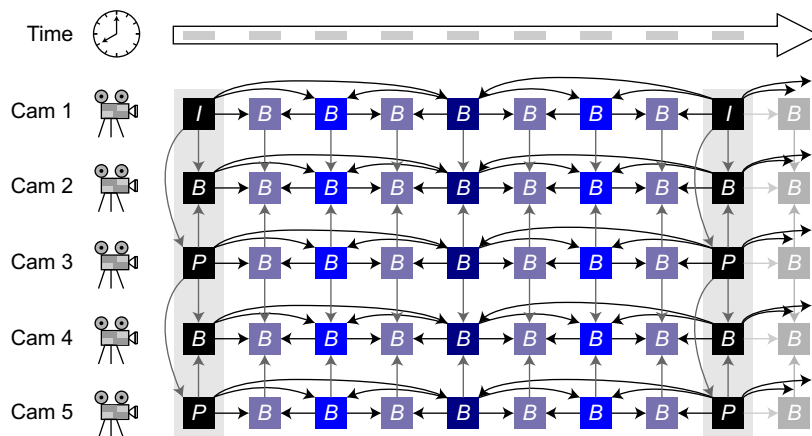


FIGURE 12.6–14

Illustration of H.264/MVC inter-view prediction hierarchy. (from public MPEG site, <http://mpeg.chiariglione.org/>)

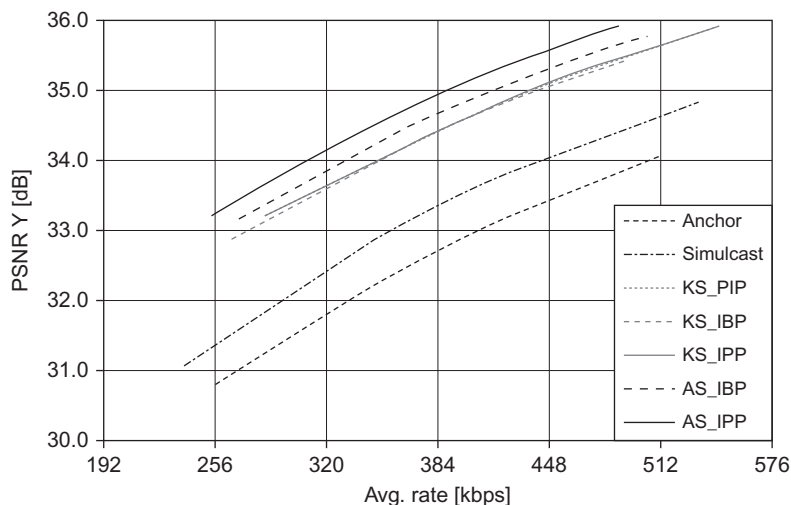


FIGURE 12.6–15

H.264/MVC result from Merkle et al. [59], © IEEE 2007.

Figure 12.6–15 are from [59] and show H.264/MVC PSNR-Y values achieved on the multiview test clip *Ballroom* consisting of eight cameras in a linear array with 19.5-cm intercamera spacing, with all the cameras pointing exactly parallel and normal to the array line, and each camera with a resolution of 640×480 pixels in 4:2:0 format at 25 fps and a sequence length 250 frames [60]. In this figure, the indicated anchor sequence is the total bitrate for conventional H.264/AVC coding in IBBP... mode of all the views. The simulcast is the same, but with hierarchical *B*-frame structure. The other curves marked AS_XXX and KS_XXX correspond to various strategies for MVC using the preceding method. We can see a clear advantage for the joint or MVC coding, amounting to several dB at CBR or about 25% savings in bitrate for constant PSNR-Y. Based on usage statistics, it is found that the inter-view prediction modes are chosen about 15–20% of the time on average. One reason for the low usage is that the individual cameras cannot be exactly calibrated to match one another. Another reason is the large inter-view displacements that can occur for objects moving in the foreground. There have been some positive results for warped view interpolation [61] for improving coding performance. In this work, pairs of views are jointly warped and interpolated toward a target, and then these synthesized views are inserted into the DPB as possible alternate references. They also compensated for camera mismatch via a look-up table method. Reportedly about 15% of the bitrate was saved for comparable performance.

12.7 NONLOCAL INTRAPREDICTION

Recently, proposals have come forth for powerful new ways to perform the intra prediction step in the AVC-based coders. They generally rely on searching for matching image blocks in the NSHP past of the current frame and then using them to predict the current or target block. This is then added as an additional mode decision for intraframe coding. There are two general methods, one based on the past original image and the other based on the past coded image.

Intra-Macroblock Displacement Compensation

This idea generalizes the motion compensation in interframe coding to searching for block matches in the current frame for *I*-frame coding. A search region is defined in the NSHP past of the current block to be coded, the best match is obtained, usually with sum of absolute differences (SAD), and then the residual block is transform coded. In addition a displacement vector must be coded and transmitted. This possibility is then added as an additional mode to the nine existing H.264/AVC predictive modes. Preliminary results credited Intra-macroblock motion compensation with a 12.4% bitrate savings and an improvement of 0.5 dB for coding the CIF version of Foreman [62].

Template Matching for Intra Prediction

This concept is similar to the preceding one but avoids the need to code and transmit the displacement overhead information. The search is conducted on the previously coded data in the NSHP past, with the same search being repeated at the receiver. The search target is no longer just the block to encode, but also includes its boundary, as shown for the 2×2 case in Figure 12.7–1. The new method was called template matching. In this exemplar of the technique, a 5-pixel part of a 3×3 block is searched in the coded/decoded past and then the right-bottom 2×2 part of the best match becomes the first quadrant of the 4×4 target block to be coded. The rest of the target block is then template coded in the same manner. The computation is equivalent to search of a 5-pixel region. The idea follows an innovative approach in texture synthesis [64]. The new template-matching method was included as an additional mode for intra prediction in the H.264/AVC JM 9.6 test coder. Results were reported including the QCIF coding of Foreman at 15 fps, comparing the new technique template-matching spatial prediction (TMSP) to the H.264/AVC test coder JM 9.6. The results are shown in Figure 12.7–2, with QP values 22, 27, 32, and 37, including both 4×4 and 8×8 transforms and the CABAC entropy coding technique. We can see reductions in bitrate for equivalent PSNR ranging from a low of 3.3% at high bitrates to a high of 11.5% reduction at low bitrates. Improvement for other test clips was lower though.

An improved TMSP coder, including template match averaging, alternative template shapes, and larger templates [65], yielded further improvement in intraframe coding under H.264/AVC.

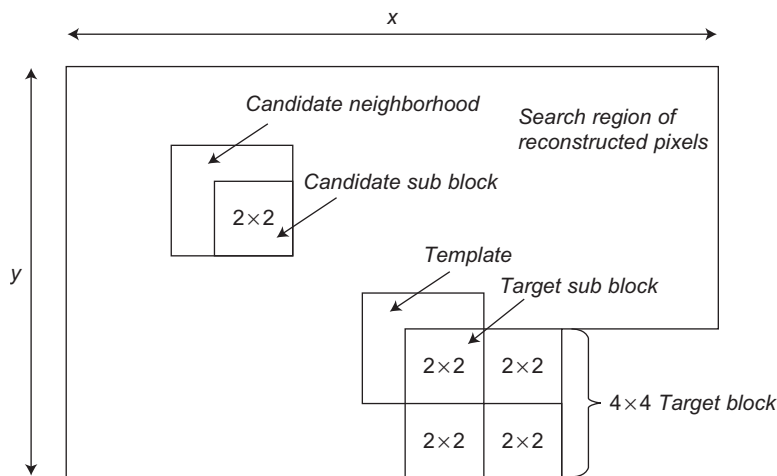


FIGURE 12.7–1

Illustration of template matching for intra prediction. (from [63])

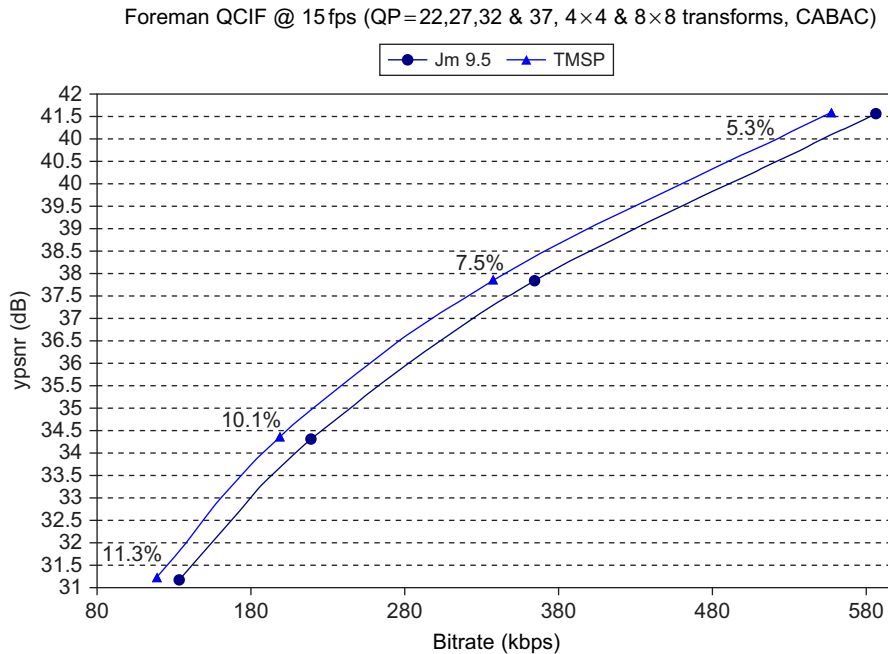


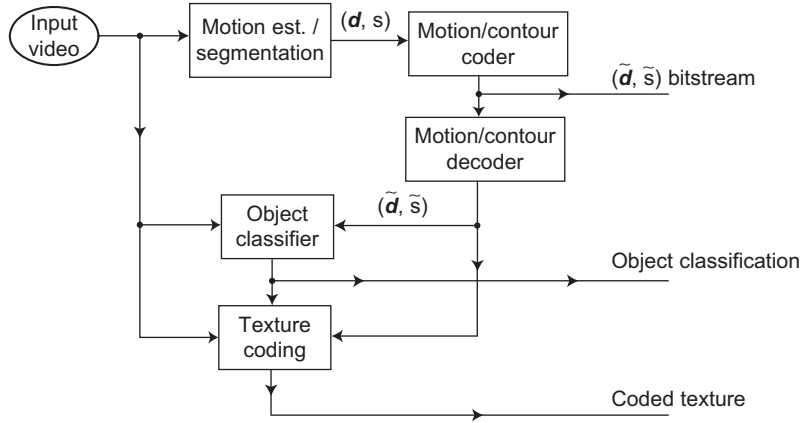
FIGURE 12.7–2

Improvement in PSNR versus bitrate for Foreman QCIF at 15 fps.

12.8 OBJECT-BASED CODING

Visual scenes are made up of objects and backgrounds. If we apply image and video analysis to pick out these objects and estimate their motion, this can be a very efficient type of video compression [66]. This was a goal early on in the MPEG 4 research and development effort, but since reliable object detection is a difficult and largely unsolved problem, the object-coding capability in MPEG 4 is largely reserved for artificially composed scenes where the objects and their locations are already known. In object-based coding, there is also the new problem of coding the outline or shape of the object, known as *shape coding*. Further, for a high-quality representation, it is necessary to allow for a slight overlap of the natural objects, or soft edge. Finally, there is the bit allocation problem, first to the object shapes, and then to their inside, called *texture*, and their motion fields.

In Chapter 11 we looked at joint Bayesian estimates of motion and segmentation. From this operation, we get both output objects and motion fields for these objects. Each one can then be subject to either hybrid MCP- or MCTF-type video coding. Here, we show an application of this Bayesian motion/segmentation

**FIGURE 12.8–1**

System diagram of the hybrid MCP object video coder in [67].

to an example object-based coder [67]. The various blocks of the object-based coder are shown in Figure 12.8–1, where we see the input video going into the joint motion estimation and segmentation block of Chapter 11, whose output is a dense motion field \mathbf{d} for each object and a segmentation label s . This output is input to the *motion/contour coder*, whose role is to code the motion field of each object and also code its contour. The input video also goes into an *object classifier* and a *texture coder*. The object classifier decides whether the object can be predicted from the previous frame, a P object, or is a new object in this frame, an I object. Of course, in the first frame, all objects are I objects. The texture coder module computes the MCP of each object and codes the prediction residual.

The motion coder approximates the dense motion field of the Bayes estimate over each object with an affine motion model fitted in a least-squares sense. An affine model with six parameters representing rotation and translation is projected onto the image plane as

$$d_1(x_1, x_2) = a_{11}x_1 + a_{12}x_2 + a_{13},$$

$$d_2(x_1, x_2) = a_{21}x_1 + a_{22}x_2 + a_{23},$$

where the position vector $\mathbf{x} = (x_1, x_2)^T$ on the object and $\mathbf{d}(\mathbf{x})$ is the displacement vector. The motion warping effect of an affine motion model has been found to well approximate the apparent motion of the pixels of rigid objects. The motion output of this coder is denoted $\tilde{\mathbf{d}}$ in Figure 12.8–1. The other output of the motion/contour coder is the shape information \tilde{s} . Since the Bayesian model enforces a continuity along the object track as well as a spatial smoothness, this information can be well coded in a predictive sense. The details are contained in Han and Woods [68].

The texture coding proceeds for each object separately. Note that these are not really physical objects, but the objects selected by the joint Bayesian MAP motion/segmentation estimate. Still, they correspond to at least large parts of physical objects. Secondly, note that it is the motion field that has been segmented not the imaged objects themselves. As a result, when an object stops moving, it gets merged into the background segment. There are now a number of nice solutions to the coding of these irregularly shaped objects. There is the shape-adaptive DCT [69] that is used in MPEG 4. There are also various SWT extensions to nonrectangular objects [70, 71].

Example 12.8–1: Object-based SWT Coder

Han [67] used the SWT method of Bernard [70] to code the *carphone* QCIF test clip was coded at the frame rate of 7.5 fps and at the CBR bitrate of 24 Kbps, and the results were compared against an H.263 implementation from Telenor Research. Figure 12.8–2 shows a decoded frame from the test clip coded at 24 Kbps via the object-based SWT coder. Figure 12.8–3 shows the same frame from the decoded output of the H.263 coder. The



FIGURE 12.8–2

QCIF of the carphone clip coded via object-based SWT at 24 Kbps.



FIGURE 12.8–3

QCIF of the carphone clip coded at 24 Kbps by H.263.

average PSNRs are 30.0 dB and 30.1 dB, respectively. So there is a very slight advantage of 0.1 dB to the H.263 coder. However, we can see that the image out of the object-based coder is sharper. A typical frame-segmentation results were given in Figure 11.4–7b in Chapter 11. Since the object-based coding was done at a constant number of bits per frame, while the H.263 coder had use of a decoded picture buffer, the latter was able to put more bits into the frames with a lot of motion, and fewer bits in quieter frames. So there was some measure of improvement possible for the object-based SWT coder. ■

A comprehensive review article on object-based video coding is contained in [72].

12.9 COMMENTS ON THE SENSITIVITY OF COMPRESSED VIDEO

Just as with image coding, the efficient variable length code words used in common video coding algorithms render them sensitive to errors in transmission. So, any usable codec must include general and repeated header information, giving positions and length of coded blocks that are separated by sync words such as EOB. Of course, additionally the decoder has to know what to do upon encountering an error (i.e., an invalid result), such as EOB in the wrong place or an illegal code-word resulting from a noncomplete VLC code tree. It must do something reasonable to recover from such a detected error. Thus, at least some redundancy must be contained in the bitstream to enable a video codec to be used in a practical environment, where even one bit may come through in error, e.g., in a storage application such as DVD or Blu-Ray disk.

Additionally, when compressed video is sent over a channel or network, it must be made robust with respect to much more frequent errors and data loss. Again, this is due to VLCs used in the compression algorithms. As we have seen, the various coding standards place data into slices or groups of blocks that terminate with a sync word, which prevents the error propagation across this boundary. For packet-based wired networks such as the Internet, the main source of error is lost packets, as errors are typically corrected in lower network layers before the packet is passed up to the application layer and given to the decoder. Because of VLC, lost packets may make further packets useless until the end of the current slice or group of blocks. Forward error correction codes can be used to protect the compressed data. Slice lengths are designed based on efficiency, burst length, and preferred network packet or cell size. Further, any remaining redundancy, after compression, can be used at the receiver for a post-decoding cleanup phase, called *error concealment* in the literature. Combinations of the two methods can be effective too. Also useful is a request for retransmission or ACK strategy, wherein a lost packet is detected at the source by not receiving an acknowledgement from the receiver and, based on its importance, may be resent. This strategy generally will require a larger output buffer and may not be suitable for real-time communication.

Scalable coding can be very effective to combat the uncertainty of a variable transmission medium such as the Internet. The ability to respond to variations in the

available bandwidth (i.e., the bitrate that can be sustained without too much packet loss) gives scalable coders a big advantage for such situations. The next chapter concentrates on video for networks.

CONCLUSIONS

Hybrid block transform methods dominate today's video coding standards. Interesting research coders have been developed based on SWT methods, showing the high degree of scalability that is inherent in this approach. Subband/wavelet schemes computationally scale nicely to super high definition SHD, close to DC resolution, where the DCI has chosen M-JPEG 2000 as their recommended standard. The current H.264/AVC standard has been extended to scalable and to multiview coding, both using an efficient hierarchical B-frame concept borrowed from MCTF. For further reading, an overview of video compression is contained in Chapter 6, "Video Compression," in Bovik's handbook [73]. See also Chapters 9–11 in *The Essential Guide to Video Processing* [50].

PROBLEMS

1. How many GBytes are necessary to store 1 hour of SD 720×486 progressive video at 8 bits/pixel and 4:2:2 color space? How much for 4:4:4? How much for 4:2:0?
2. A certain analog video display/monitor has a 30-MHz video bandwidth. Can it resolve the pixels in a 1080p video stream? Remember, 1080p is the ATSC format with 1920×1080 pixels and 60 fps.
3. The NTSC DV color space 4:1:1 is specified in Figure 12.1–5, and the NTSC MPEG 2 color space 4:2:0 is given by Figure 12.P–1.
 - (a) First, assume that each format is progressive. What is the specification for the chroma sample rate change required? Specify the upsampler, the ideal filter specifications, and the downsampler.
 - (b) Do the same for an interlace format noting that the chroma samples are part of the top field only, as shown in Figure 12.P–2.

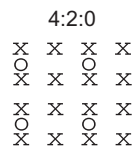


FIGURE 12.P–1

4:2:0 color space structure of MPEG 2.

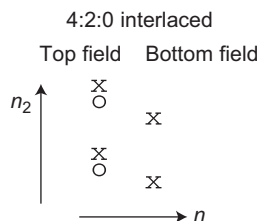


FIGURE 12.P-2

4:2:0 chrominance samples are part of top field only.

4. Use a Lagrangian formulation to optimize the intraframe coder with distortion $D(n)$ as in (12.1-4), and average bitrate $R(n)$ in (12.1-5), both for frame n . Consider frames 1 to N and find an expression for the resulting average mean-square distortion as a function of the average bitrate.
5. Write the decoder diagram for the backward motion-compensated hybrid encoder shown in Figure 12.2-4.
6. Discuss the algorithmic (structural) delay of an MPEG hybrid coder based on the number of B frames that it uses. What is the delay when the successive number of B frames is M ?
7. In an MPEG 2 coder, let there be two B frames between each non- B frame, and let the GOP size be 15. At the source, we denote the frames in one GOP as follows:

I_1	B_2	B_3	P_4	B_5	B_6	P_7	B_8	B_9	P_{10}	B_{11}	B_{12}	P_{13}	B_{14}	B_{15}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

 - (a) What is the order in which the frames must be coded?
 - (b) What is the order in which the frames must be decoded?
 - (c) What is the required display order?
 - (d) Based on the preceding answers (a-c), what is the inherent delay in this form of MPEG 2? Can you generalize your result from two successive B frames to M successive B frames?
8. In MPEG 1 and 2, there are *open* and *closed* GOPs. In the closed GOP case, the bidirectional predictor cannot use any frames outside the current GOP in calculating B frames. In the open GOP case, they can be used. Consider a GOP of 15 frames, using pairs of B frames between each P frame. Which frames would be affected by the difference? How would the open GOP concept affect decodability?
9. Find the synthesis filter corresponding to the lifting analysis equations (12.4-1) and (12.4-2). Is the reconstruction exact even in the case of subpixel accuracy? Why? Do the same for the LGT 5/3 SWT, as in (12.4-3) and (12.4-4).

10. In an MCTF-based video coder, unconnected pixels can arise due to expansion and contraction of the motion field but also, and more importantly, due to occlusion. State the difference between these two cases and discuss how they should be handled in the coding stage that comes after the MCTF.
11. Find the “inverse” transform for the 4×4 DCT-like matrix (12.6–1) used in H.264/AVC. Note that the “inverse” transform used in H.264/AVC is *not* the matrix inverse, but rather a kind of scaled version of the inverse. Show that 8-bit data may be transformed and inverse transformed without any error using 16-bit arithmetic. Refer to [43] or [14] for help.
12. String the frames in Figure 12.6–14 together so that the indicated MVC coding can be accomplished by a standards-conforming H.264/AVC coder.

REFERENCES

- [1] P. H. N. de With and A. M. A. Rijckaert, “Design Considerations of the Video Compression System of the New DV Camcorder Standard,” *IEEE Trans. Consumer Electr.*, vol. 43, no. 4, pp. 1160–1179, November 1997.
- [2] C. B. Jones, “An Efficient Coding System for Long Source Sequences,” *IEEE Trans. Inform. Theory*, vol. IT-27, no. 3, pp. 280–291, May 1981.
- [3] T. Naveen and J. W. Woods, “Subband and Wavelet Filters for High-Definition Video Compression,” Chapter 8 in *Handbook of Visual Comm.*, Ed. H.-M. Hang and J. W. Woods, Academic Press, 1995.
- [4] J. W. Woods, S.-C. Han, S.-T. Hsiang, and T. Naveen, “Spatiotemporal Subband/Wavelet Video Compression,” Chapt. 6.2 in *Handbook of Image and Video Process.*, 1st Ed., Al Bovik, Ed., pp. 575–595, Academic Press, 2000.
- [5] J. C. Candy et al., “Transmitting Television as Clusters of Frame-to-Frame Differences,” *Bell Syst. Tech. J.*, vol. 50, pp. 1889–1917, July–August 1971.
- [6] A. Netravali and J. Limb, “Picture Coding: A Review,” *Proceedings of the IEEE*, no. 3, March 1980.
- [7] B. Girod, “The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences,” *IEEE J. Select. Areas in Comm.*, vol. SAC-5, no. 7, pp. 1140–1154, August 1987.
- [8] *Generic Coding of Motion Pictures and Assoc. Audio*, Rec. H.262 (aka MPEG-2), ISO/IEC 13818-2, March 1994.
- [9] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman and Hall, New York, 1997.
- [10] E.-H. Yang and L. Wang, “Full Rate Distortion Optimization of MPEG-2 Video Coding,” *Proc. ICIP 2009*, Cairo, Egypt, November 2009.
- [11] *Coded Representation of Picture and Audio Information—MPEG-2 Test Model 5*, ISO-IEC AC-491, April 1993.
- [12] K. P. Davies, “HDTV Evolves for the Digital Era,” in HDTV mini issue of *IEEE Communications Magazine*, vol. 34, no. 6, pp. 110–112, June 1996.
- [13] F. Pereira and T. Ebrahimi, Eds., *The MPEG-4 Book*, Prentice-Hall, Inc. Upper Saddle River, NJ, 2002.

- [14] S. Liu and A. Bovik, "Digital Video Transcoding," Chapter 6.3, in *Handbook of Imaging and Video Processing*, 2nd Ed., A. Bovik, ed., Elsevier/Academic Press, Burlington, MA, 2005.
- [15] G. D. Karlson, *Subband Coding for Packet Video*, MS thesis, Center for Telecommunications Research Columbia University, 1989.
- [16] D. LeGall and A. Tabatabai, "Sub-band Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques," *Proc. ICASSP 1988*, IEEE, New York, pp. 761–764, April 1988.
- [17] W. Glenn et al., "Simple Scalable Video Compression Using 3-D Subband Coding," *SMPTE Journal*, March 1996.
- [18] J. W. Woods and T. Naveen, "Subband Coding of Video Sequences," *SPIE Proc. VCIP*, SPIE, pp. 724–732, November 1989.
- [19] T. Naveen and J. W. Woods, "Motion Compensated Multiresolution Transmission of Video," *IEEE Trans. Video Tech.*, vol. 4, no. 1, pp. 29–43, February 1994.
- [20] H. Gharavi, "Motion Estimation Within Subbands," Chapt. 6 in *Subband Image Coding*. Ed. J. W. Woods, Kluwer Academic Pub, 1991.
- [21] L. Vandendorpe, *Hierarchical Coding of Digital Moving Pictures*, PhD thesis, University of Louvain, Belgium, 1991.
- [22] F. Bosveld, *Hierarchical Video Compression Using SBC*, PhD thesis, TU Delft, The Netherlands, 1996.
- [23] T. Kronander, "Motion Compensated 3-Dimensional Waveform Image Coding," *Proc. ICASSP*, IEEE, Glasgow, Scotland, 1989.
- [24] J.-R. Ohm, "Three-dimensional Subband Coding with Motion Compensation," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 559–571, September 1994.
- [25] S.-J. Choi, *Three-Dimensional Subband/Wavelet Coding of Video*, PhD thesis, ECSE Department, Rensselaer Polytechnic, Troy, NY, August 1996.
- [26] T. Naveen and J. W. Woods, "Subband Finite-State Scalar Quantization," *IEEE Trans. Image Processing*, vol. 5, no. 1, pp. 150–155, January 1996.
- [27] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Scalable Video Coding and Transport over Broadband Wireless Networks," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 6–20, January 2001.
- [28] A. W. Johnson, T. Sikora, T. K. Tan, and K. N. Ngan, "Filters for Drift Reduction in Frequency Scalable Video Coding Schemes," *Electron. Letters*, vol. 30, no. 6, pp. 471–472, March 1994.
- [29] P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.*, vol. 31, no. 4, pp. 532–540, April 1983.
- [30] T. Naveen, F. Bosveld, R. Lagendijk, and J. W. Woods, "Rate Constrained Multiresolution Transmission of Video," *IEEE Trans. Video Tech.*, vol. 5, no. 3, pp. 193–206, June 1995.
- [31] D. Taubman and A. Zakhori, "Multirate 3-D Subband Coding of Video," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 572–588, September 1994.
- [32] J. W. Woods and G. Lilienfeld, "Resolution and Frame-rate Scalable Video Coding," *IEEE Trans. Video Tech.*, vol. 11, no. 9, pp. 1035–1044, September 2001.
- [33] S.-T. Hsiang and J. W. Woods, "Embedded Video Coding Using Invertible Motion Compensated 3-D Subband/Wavelet Filter Bank," *Signal Processing: Image Communication*, vol. 16, no. 8, pp. 705–724, May 2001.
- [34] S.-T. Hsiang and J. W. Woods, "Embedded Image Coding Using Zeroblocks of Subband/Wavelet Coefficients and Context Modeling," *MPEG-4 Workshop and Exhibition at ISCAS 2000*, IEEE, Geneva, Switzerland, May 2000.

- [35] S. Choi and J. W. Woods, "Motion-compensated 3-D Subband Coding of Video," *IEEE Trans. Image Process.*, vol. 8, no. 2, pp. 155–167, February 1999.
- [36] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with Lifting Implementation," *IEEE Trans. Video Tech.*, vol. 14, no. 10, pp. 1183–1194, October 2004.
- [37] J. Ohm, M. vd Schaar, and J. W. Woods, "Interframe Wavelet Coding—Motion Picture Representation for Universal Scalability," *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 877–908, October 2004.
- [38] Y. Wu, K. Hanke, T. Russert, and J. W. Woods, "Enhanced MC-EZBC Scalable Video Coder," *IEEE Trans. Video Tech.*, vol. 18, no. 10, pp. 1432–1436, October 2008.
- [39] T. Russert, K. Hanke, and M. Wien, "Optimization for Locally Adaptive MCTF Based on 5/3 Filtering," *Proc. Picture Coding Symposium (PCS)*, San Francisco, CA, 2004.
- [40] R. Xiong, J. Xu, F. Wu, S. Li, and Y. Zhang, "Layered Motion Estimation and Coding for Fully Scalable 3-D Wavelet Video Coding," in *Proc. ICIP*, Singapore, pp. 2271–2274, October 2004.
- [41] A. Secker and D. Taubman, "Highly Scalable Video Compression with Scalable Motion Coding," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1029–1041, August 2004.
- [42] G. J. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Process. Magazine*, vol. 15, no. 6, pp. 74–90, November 1998.
- [43] G. J. Sullivan and T. Wiegand, "Video Compression—From Concepts to the H.264/AVC Standard," *Proc. of the IEEE*, vol. 93, no. 1, pp. 18–31, January 2005.
- [44] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Video Tech.*, vol. 13, no. 7, pp. 560–576, July 2003.
- [45] Special Issue on H.264/AVC, *IEEE Trans. for Video Tech.*, vol. 13, no. 7, July 2003.
- [46] T. Wiegand and B. Girod, "Lagrange Multiplier Selection in Hybrid Video Coder Control," *Proc. IEEE ICIP*, 2001.
- [47] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," *IEEE Trans. Video Tech.*, vol. 13, no. 7, pp. 688–703, July 2003.
- [48] E.-H. Yang and X. Yu, "Rate Distortion Optimization for H.264 Interframe Coding: A General Framework and Algorithms," *IEEE Trans. Image Process.*, vol. 16, no. 7, pp. 1774–1784, July 2007.
- [49] *H.264 reference software*. Available at <http://iphome.hhi.de/suehring/ttml/>.
- [50] Al Bovik, Ed., *The Essential Guide to Video Processing*, Elsevier Academic Press, Burlington, MA, 2009.
- [51] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. Video Tech.*, vol. 17, no. 9, pp. 1103–1120, September 2007.
- [52] H. Schwarz and D. Marpe, "Analysis of Hierarchical B Pictures and MCTF," *Proc. Int. Conf. Multimedia Expo. (ICME)*, Toronto, Canada, July 2006.
- [53] H. Schwarz and T. Wiegand, *Further Results for an RD-Optimized Multi-loop SVC Encoder*, JVT-W072, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 23rd Meeting: San Jose, CA, April 2007.
- [54] H. Schwarz and M. Wien, "The Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 135–141, March 2008.
- [55] C. Segall and G. Sullivan, "Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension," *IEEE Trans. Video Tech.*, vol. 17, no. 9, pp. 1121–1135, September 2007.
- [56] S. Wenger, "Video Redundancy Coding in H.263+," *Proc. Workshop on Audio-Visual Services for Packet Networks*, 1997.

- [57] M. R. Civanlar, R. D. Gaglinello, and G. L. Cash, "Efficient Multi-Resolution, Multi-Stream Video Systems Using Standard Codecs," *Journal of VLSI Signal Processing*, vol. 17, no. 2–3, pp. 269–279, 1997.
- [58] A. Eleftheriadis, M. R. Civanlar, and O. Shapiro, "Multipoint Videoconferencing with SVC," *Journal of Zhejiang University-Science A*, vol. 7, no. 5, pp. 696–705, May 2006.
- [59] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Efficient Prediction Structures for Multiview Video Coding," *IEEE Trans. Video Tech.*, vol. 17, no. 11, pp. 1461–1473, November 2007.
- [60] MERL test clips for MVC. Available at <ftp://ftp.merl.com/pub/avetro/mvc-testseq>.
- [61] K. Yamamoto, M. Kitahara, H. Kimata, T. Yendo, T. Fujii, M. Tanimoto, S. Shimizu, K. Kamikura, and Y. Yashima, "Multiview Video Coding Using View Interpolation and Color Correction," *IEEE Trans. Video Tech.*, vol. 17, no. 11, pp. 1436–1449, November 2007.
- [62] S.-L. Yu and C. Chrysafis, "New Intra Prediction Using Intra-Macroblock Motion Compensation," *Joint Video Team of MPEG and VCEG, JVT-C151*, Fairfax, VA, May 2002.
- [63] T.-K. Tan, C. S. Boon, and Y. Suzuki, "Intra Prediction by Template Matching," *Proc. IEEE ICIP*, Atlanta, GA, October 2006.
- [64] L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. ACM SIGGRAPH*, New Orleans, LA, 2000.
- [65] T.-K. Tan, C. S. Boon, and Y. Suzuki, "Intra Prediction by Averaged Template Matching Predictors," *Proc. IEEE CCNC*, 2007.
- [66] H. Musmann, M. Hotter, and J. Ostermann, "Object-Oriented Analysis-Synthesis Coding of Moving Images," *Signal Process.: Image Communications*, vol. 1, no. 2, pp. 117–138, October 1989.
- [67] S.-C. Han, *Object-Based Representation and Compression of Image Sequences*, PhD thesis, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 1997.
- [68] S.-C. Han and J. W. Woods, "Adaptive Coding of Moving Objects for Very Low Bitrates," *IEEE J. Select. Areas in Comm.*, vol. 16, no. 1, pp. 56–70, January 1998.
- [69] T. Sikora, "Low-Complexity Shape-Adaptive DCT for Coding of Arbitrarily Shaped Image Segments," *Signal Process.: Image Comm.*, vol. 7, no. 4–6, pp. 381–395, 1995.
- [70] H. J. Bernard, *Image and Video Coding Using a Wavelet Decomposition*, PhD thesis, Delft University of Technology, The Netherlands, 1994.
- [71] J. Li and S. Lei, "Arbitrary Shape Wavelet Transform with Phase Alignment," *Proc. ICIP*, vol. 3, pp. 683–687, 1998.
- [72] T. Ebrahimi and M. Kunt, "Object-Based Video Coding," Chapter 6.3 in *Handbook of Image and Video Processing*, A. Bovik, Ed., 1st Ed., Academic Press, 2000.
- [73] Chapter 6 in *Handbook of Image and Video Process.*, 2nd Ed., Al Bovik, Ed., Elsevier/Academic Press, Burlington, MA, 2005.

Video Transmission over Networks

13

Many video applications involve transmission over networks, such as visual conversation, video streaming, *video on demand* (VOD), and video downloading. Various kinds of networks are involved, wireless local area networks (WLANs) such as IEEE 802.11, *storage area networks* (SANs), private internet protocol (IP) networks, public Internet, and wireless networks. If the network lacks congestion and has no bit errors, then the source coding bitstreams of the previous chapter can be sent directly over the network. Otherwise, some provision has to be made to protect the rather fragile compressed data that results from the source coder. In some cases, even a single bit error can cause the decoder to terminate upon encountering illegal parameter values.

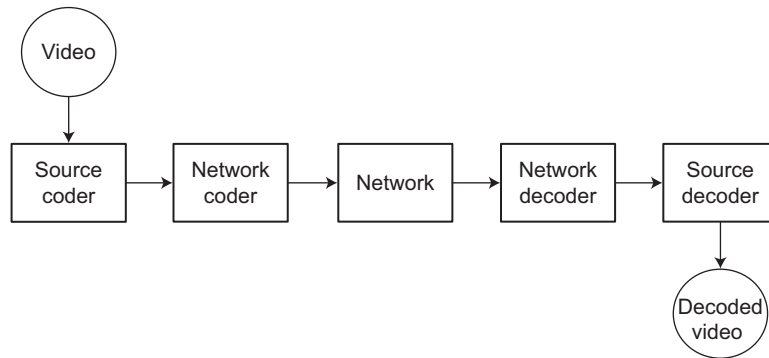
In this chapter we will encounter some of the new aspects that arise with network transmission of video. We will review the error-resilient features of some standard video source coders as well as the transport error protection features of networks and show how they can work together to protect the video data. We present some robust transmission methods for scalable SWT-based coders such as MC-EZBC as well as for standard H.264/AVC. The chapter ends with a section on joint source–network coding, wherein the source-channel coding is continued into the network on overlay nodes that also support data item adaptation. This section includes an introduction to network coding focusing on use of random network codes.

We start with an overview of IP networks. General introductions to networks themselves can be found in the textbooks by Tenenbaum [1] and Kurose and Ross [2].

13.1 VIDEO ON IP NETWORKS

This section introduces the problem of transmitting video on IP networks such as the public Internet. We first provide an overview of such networks. Then we briefly introduce the error-resilience features found on common video coders. Finally, we consider the so-called *transport-level* coding done at the network level to protect the resulting error-resilient coded video data.

A basic system diagram for networked transmission of video on a single path is shown in Figure 13.1–1. In this case the *source coder* will usually have its error-resilience features turned on and the *network coder* will exploit these features at the

**FIGURE 13.1–1**

System diagram of networked video transmission.

transport level. The network coder will generally also exploit available feedback on congestion in the network in its assignment of data to network packets, and choice of FEC or ACK/NACK strategy, to be discussed later in this chapter. This diagram is for *unicast*; thus there is just one receiver for the one video source shown here. *Multicast*, or general *broadcast*,¹ introduces additional issues, one of which is the need for scalability due to the expected heterogeneity of multicast receivers and link qualities (i.e., usable bandwidth, packet loss, delay and delay variation (jitter), and bit error rate). While we do not address multicast here, we do look at the scalability-related problem of digital item adaptation or transcoding later in this chapter.

Overview of IP Networks

Our overview of IP networks starts with the basic concept of a *best-efforts* network. We briefly introduce the various network levels, followed by an introduction to the *transfer control protocol* (TCP) and the *real-time protocol* (RTP) for video. Since the majority of network traffic is TCP, it is necessary that video transmissions over RTP be what is called TCP-friendly, meaning that, at least on the average, each RTP flow uses a fraction of network resources similar to that of a TCP flow. We then discuss how these concepts are used in packet loss prevention.

Best-Efforts Network

Most IP networks, including the public Internet, are best-efforts networks, meaning that there are no guarantees of specific performance, only that the network protocol is fair and makes a best effort to get the packets to their destination, perhaps in a somewhat reordered sequence with various delays. At intermediate nodes in the network, packets are stored at buffers of finite size that can overflow. When such overflow

¹The term “broadcast” may mean different things in the video coding community and the networks community. Here, we simply mean a very large-scale multicast to perhaps millions of subscribers, not a wholesale flooding of the entire network.

occurs, then the packet is lost, and this packet loss is the primary error that occurs in modern wired IP networks. So these networks cannot offer a guaranteed *quality of service* (QoS) such as is obtained on the private switched networks T1, T2, or ATM. However, significant work of the Internet Engineering Task Force (IETF) has concentrated on *differentiated services* (DiffServ) [3], wherein various priorities can be given to classes of packet traffic.

Layers and Protocol Stack

Figure 13.1–2a shows the open system interconnection (OSI) network layers, also called a protocol stack [1]. The lowest level is the *physical layer*, called layer 1, the level of bits. At level 2, the *data link layer*, the binary data are aggregated into *frames*, while the next higher layer 3, the *network layer*, deals only with packets. In the case of bit errors that cannot be corrected, a packet is rejected at the *data link layer* and so not passed up to the network level. However, bit errors are significant only in wireless networks. For our purposes we can skip the presentation and session layers and go directly to the *application layer* (see Figure 13.1–2b), where our video coders and decoders sit. On transmission the application sends a message down the protocol stack, with each level requesting services from the one below. On reception, the information is passed back up the stack. Hopefully, the original message is then reconstructed.

The presentation and session layers are not mentioned in [2], compressing the stack to five layers, typical in Internet discussions. The IP works in the network layer. The TCP works in the transport layer. The IP layer offers no reliability features. It just packetizes the data, adds a destination header, and sends it on its way. But the IP protocol at the network layer is responsible for all the routing or path selection to get from one node to another, all through the network to the final addressee node.

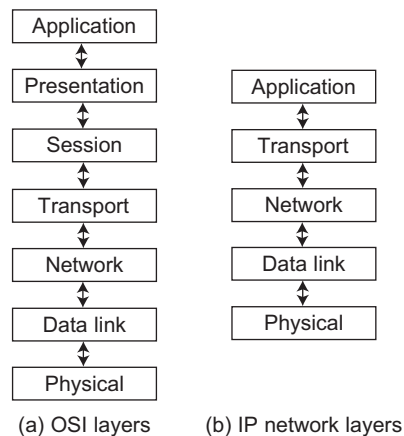


FIGURE 13.1–2

Network reference models (stack).

TCP at Transport Level

The extremely reliable TCP controls the main load of the Internet traffic. It attains the reliability needed for general computer data by retransmitting any lost IP packets, thus incurring a possibly significant increased delay upon packet loss. Most media traffic does not require such high reliability, but conversely cannot tolerate excessive variation in time delay, so-called *time jitter*. The ubiquitous TCP also contains congestion-avoidance features, which heavily modulate its sending rate to avoid packet loss, signaled to the sender by small backward-flowing control packets. TCP flows start up slowly and upon encountering significant packet loss, the sending rate is drastically reduced (by half) through a process called *additive-increase multiplicative-decrease* [2]. This increase and decrease is conducted at network epochs as determined by the so-called *round trip time* (RTT), the time it takes a packet to make a round trip from source to receiver.

RTP at the Application Layer

For real-time transmission, RTP does not employ TCP at all. The RTP protocol operates at the application layer and uses the *user datagram protocol* (UDP), which unlike TCP simply employs single IP transmissions with no checking for ACKs and no retransmissions. Each UDP packet or datagram is sent separately, with no throttling down of rate. Thus video sent via RTP will soon dominate the flows on a network unless some kind of rate control is applied.

TCP Friendly

The flow of video packets via RTP/UDP must be controlled so as not to dominate the bulk of network traffic, which flows via TCP. Various TCP-friendly rate equations have been devised [4, 5] that have been shown to use on average a nearly equal amount of network bandwidth as TCP. One such equation [5] is

$$R_{\text{TCP}} = \frac{MTU}{RTT \sqrt{\frac{2p}{3}} + RTO \sqrt{\frac{27p}{8}} p(1 + 32p^2)}, \quad (13.1-1)$$

where *RTT* is the round trip time, *RTO* is the receiver time-out time (i.e., the time that a sender can wait for the ACK before it declares the packet lost), *MTU* is the maximum transmission unit (packet) size, and *p* is the packet loss probability, also called *packet-loss rate*. If the RTP video sender follows this equation, then its sending rate will be TCP friendly and should avoid being a cause of network congestion.

As mentioned previously, the main loss mechanism in wired IP networks is packet overflow at the *first-in first-out* (FIFO) internal network buffers. If the source controls its sending rate according to (13.1-1), then its average packet loss should be approximately that of a TCP flow, which is low at 3–6% [6]. So control of the video sending rate offers two benefits, one to the sender and one to the network. Bit errors are not considered a problem in wired networks. Also, any possible small number of bit errors is detected and not passed up the protocol stack, so from the application's viewpoint, the packet is lost. In the wireless case, bit errors are important equally with packet loss.

Error-Resilient Coding

Error-resilient features are often added to a video coder to make it more robust to channel errors. Here, we discuss data partitioning and slices that are introduced for purposes of re-synchronization. We also briefly consider reversible VLCs, which can be decoded starting from either the beginning or the end of the bitstream. We introduce the multiple description feature, whereby decoding at partial quality is still possible even if some of the descriptions (packets) are lost.

Data Partitioning and Slices

If all the coded data is put into one partition, then a problem in transmission can cause the loss of the entire remaining data, due first to the use of VLCs and then to the heavy use of spatial and temporal prediction in video coders. In fact if synchronization is lost somewhere, it might never be regained without the insertion of *sync words*² into the bitstream. In MPEG 2, *data partition* (DP) consisted of just two partitions on a block basis: one for the motion vector and low frequency DCT coefficients, and a second one for high frequency coefficients [7]. Similar to DP is the concept of *slice*, which breaks a coded frame up into separate coded units, wherein reference cannot be made to other slices in this frame. Slices start (or end) with a sync word, and contain position, length of slice in pixels, and other needed information in a header, permitting the slice to be decoded independent of prior data. But because slices cannot refer to other slices in the same frame, compression efficiency suffers somewhat.

Reversible Variable Length Coding

Variable length codes such as Huffman are constructed on a tree and thus satisfy a *prefix condition*, which makes them uniquely decodable (i.e., no codeword is the prefix of another codeword). Also, since the Huffman code is optimal, the code tree is fully populated (i.e., all leaf nodes are valid messages). If we lose a middle segment in a string of variable length coding (VLC) codewords, in the absence of other errors, we can only do correct (forward) decoding from the beginning of the string up to the lost segment. However, at the error location we lose track and will probably not be able to correctly decode the rest of the string, after the lost segment. On the other hand, if the VLC codewords satisfy a *suffix condition*, then we could start at the end of the string and decode back to the lost segment, and thus recover the lost data. Now as mentioned in Chapter 9, most Huffman codes cannot satisfy both a suffix and a prefix condition, but methods that find *reversible VLCs* (RVLCs) starting from a conventional VLC have been discovered [8]. Unfortunately, the resulting RVLCs lose significant efficiency versus Huffman codes.

Regarding structural VLCs, the Golomb-Rice codes are near-optimal codes for the geometric distribution, and the Exp-Golomb codes are near-optimal for distributions more peaky than exponential that can occur in the prediction residuals of

²A sync word is a long and unique codeword that is guaranteed not to occur naturally in the coded bitstream. See end-of-chapter problem 1.

video coding. RVLCs with the same length distribution as Golomb-Rice and Exp-Golomb were discovered by Wen and Villasenor [9], and Exp-Golomb codes are used in the video coding standard H.264/AVC. Also in Farber et al. [10] and Girod [11], a method is presented that achieves bidirectional decoding with conventional VLCs. In this method, forward and reverse codestreams are combined with a delay for the maximum length codeword, and the compression efficiency is almost the same as for Huffman coding when the string of source messages is long.

Multiple Description Coding

The main concept in *multiple description coding* (MDC) is to code the video data into two or more multiple streams that each carry a near-equal amount of information. If all the streams are received, then decoding proceeds normally. If fewer descriptions are received, then the video can still be decoded, but at a reduced quality. Also, the quality should be almost independent of which descriptions are received [12]. If we now think of the multiple descriptions as packets sent out on a best-efforts network, we can see the value of an MDC coding method for networked video.

Example 13.1–1: Scalar MDC Quantizer

Consider an example where we have to quantize a scalar random variable X whose range is $[1, 10]$, with a quantizer having 10 equally spaced output values: $1, 2, \dots, 9, 10$. Call this the original quantizer. We can also accomplish this with two separate quantizers, called *even quantizer* and *odd quantizer*, each with step size 2, that output even and odd representation values, respectively. Clearly, if we receive either even or odd quantizer output, we can reconstruct the value of X up to an error of ± 1 , and if we receive both even and odd quantizer outputs, then this is the same as receiving the original quantizer output

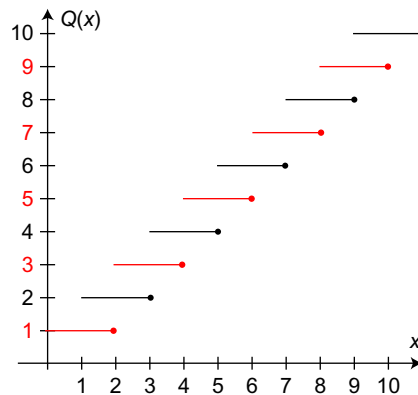


FIGURE 13.1–3

Illustration of an MDC scalar quantizer.

having 10 levels, so that the maximum output error is now ± 0.5 . In Figure 13.1–3 the even quantizer output levels are denoted by the gray lines and the odd quantizer output levels are denoted by the black lines. We can see a redundancy in the two quantizers, because the gray and black horizontal lines overlap horizontally. ■

One simple consequence we see from this example is that the MDC method introduces a redundancy into the two (or more) codestreams or descriptions. It is this redundancy that protects the data against loss of one or another bitstream. The concept of MDC has been extended far beyond simple scalar quantizers, including MD transform coding [13] and the MD-FEC coding [14], that we will apply to a layered codestream later in this chapter.

Transport-Level Error Control

The preceding error-resilient features can be applied in the video coding or application layer. Then the packets are passed down to the *transport level* for transmission over the channel or network. We look at two powerful and somewhat complementary techniques, error control coding and acknowledgement-retransmission (ACK/NAK). Error control coding is sometimes called *forward error correction* (FEC) because only a forward channel is used. However, in a packet network there is usually a backward channel, so that acknowledgments can be fed back from receiver to transmitter, resulting in the familiar ACK/NAK signal. Using FEC we must know the present channel quality fairly well or risk wasting error control (parity) bits on the one hand, or not having enough parity bits to correct the error on the other hand. In the simpler ACK/NAK case, we do not compromise the forward channel bandwidth at all and only transmit on the backward channel a very small ACK/NAK packet, but we do need the existence of this backward channel. In delay-sensitive applications like visual conferencing, we generally cannot afford to wait for the ACK and the subsequent retransmission, due to stringent total delay requirements (≤ 250 msec [15]). Some data “channels” where there is no backward channel are the CD, the DVD, and TV broadcast. There is a back channel in Internet unicast, multicast, and broadcast. However, in the latter two, multicast and broadcast, there is the need to consolidate the user feedback at overlay nodes to make the system scale to possibilities of large numbers of users.

Forward Error Control Coding

The FEC technique is used in many communication systems. In the simplest case, it consists of a block coding wherein a number $n - k$ of parity bits are added to k binary information bits to create a binary channel codeword of length n . The Hamming codes are examples of binary linear codes, where the codewords exist in n -dimensional binary space. Each code is characterized by its minimum Hamming distance d_{\min} , defined as the minimum number of bit differences between two different codewords. Thus, it takes d_{\min} bit errors to change one codeword into another. So the error detection capability of a code is $d_{\min} - 1$, and the error correction capability of a code

is $\lfloor d_{\min}/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the least integer function. This last is so because if fewer than $\lfloor d_{\min}/2 \rfloor$ errors occur, the received string is still closer (in Hamming distance) to its error-free version than to any other codeword. Reed-Solomon (RS) codes are also linear, but operate on symbols in a so-called Galois field with 2^l elements. Codewords, parity words, and minimal distance are all computed using the arithmetic of this field. An example is $l = 4$, which corresponds to hexadecimal arithmetic with 16 symbols. The $(n, k) = (15, 9)$ RS code has hexadecimal symbols and can correct 3 symbol errors. It codes 9 hexadecimal information symbols (36 bits) into 15 symbol codewords (60 bits) [16]. The RS codes are perfect codes, meaning that the minimum distance between codewords attains the maximum value $d_{\min} = n - k + 1$ [17]. Thus an (n, k) RS code can detect up to $n - k$ symbol errors. The RS codes are very good for bursts of errors since a short symbol error burst translates into an l times longer binary error burst, when the symbols are written in terms of their l -bit binary code [18]. These RS codes are used in the CD and DVD standards to correct error bursts on decoding.

Automatic Repeat Request

A simple alternative to using error control coding is the *automatic repeat request* (ARQ) strategy of acknowledgement and retransmission. It is particularly attractive in the context of an IP network and is used exclusively by TCP in the transport layer. There is no explicit expansion needed in available bandwidth, as would be the case with FEC, and no extra congestion, unless a packet is not acknowledged (i.e., no ACK is received). TCP has a certain *timeout* interval [2], at which time the sender acts by retransmitting the unacknowledged packet. Usually the timeout is set to be larger than the RTT. (Note that this can lead to duplicate packets being received under some circumstances.) At the network layer, the IP protocol has a header check sum that can cause packets to be discarded there. While ARQ techniques typically result in too much delay for visual conferencing, they are quite suitable for video streaming, where playout buffers are typically 5 seconds or more in length.

Wireless Networks

The situation in video coding for wireless networks is less settled than that for the usually more benign wired case. In a well-functioning wired IP network, usually bit errors in the cells (packets) can almost be ignored, being of quite low probability, especially at the application layer. In the wireless case, the SNR is highly variable, with bit errors occurring in bursts and at a much higher rate. Methods for transporting video over such networks must consider both packet loss and bursty bit errors.

The common protocol TCP is not especially suitable for the wireless network because it interprets packet loss as indicating congestion, while in a wireless network packet loss may be largely due to bit errors, in turn causing the packet to be deleted at the link layer [2]. Variations of TCP have been developed for wireless transmission [19], one of which is to just provide for link layer retransmissions. So-called

cross-layer techniques have become popular [20] for exploiting those packets with bit errors. A simple cross-layer example is that packets with detected errors could be sent up to the application level instead of being discarded. There, RVLCs can be used to decode the packet from each end until reaching the location where the error was detected, thus recovering much of the loss.

Joint Source-Channel Coding

A most powerful possibility is to design the video coder and channel coder together to optimize overall performance in the operational rate-distortion sense. If we add FEC to the source coder bitrate, then we have a higher channel bitrate. For example, a channel rate $R_C = 1/2$ Reed-Solomon (RS) code would double the channel bitrate. This channel bitrate translates into a probability of bit error. Depending on the statistics of the channel (i.e., independent bit errors or bursts), the bit error can be converted to a word error and frame error, where a frame would correspond to a packet, which may hold a slice or a number of slices. Typically these packets with bit errors will not be passed up to the application layer and will be considered lost packets. Finally, considering error concealment provisions in the decoder, this packet-loss rate can translate into an average PSNR for the decoded video frames. Optimization of the overall system is called *joint source channel coding* (JSCC), and essentially consists of trading off the source coder bitrate versus FEC parity bits for the purpose of increasing the average decoder PSNR. Modestino and Daut [21] published an early paper introducing JSCC for image transmission.

Bystrom and Modestino [22] introduced the idea of a universal distortion-rate characteristic for the JSCC problem. Writing the total distortion from source and channel coding as D_{S+C} , they express this total distortion as a function of inverse bit-error probability on the channel p_b for various source coding rates R_S . A sketch of such a function is shown in Figure 13.1–4. This function can be obtained either analytically or via simulation for the source coder in question, but precomputing such

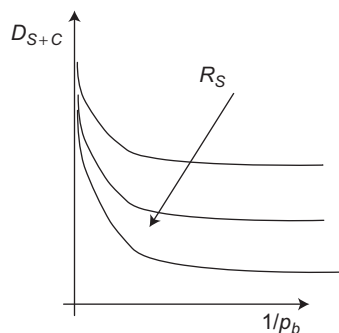


FIGURE 13.1–4

Total distortion plotted versus reciprocal of channel bit error probability, with source coding rate as a parameter.

a curve can be computationally demanding. The characteristic does not depend on channel coding, modulation, or channel model. While the method does assume that successive bit errors are independent, the authors in [22] state that this situation can be well approximated by standard interleaving methods.

This universal distortion-rate characteristic of the source can be combined with a standard bit-error probability curve of the channel, as seen in Figure 13.1–5, to obtain a plot of D_{S+C} versus either source rate R_S (source bits/pixel) or what the authors in [22] call $R_{S+C} \triangleq R_S/R_C$, where R_C is the channel coding rate, expressed in terms of source bits divided by total bits (i.e., source bits plus parity bits). The units of R_{S+C} are then bits/c.u., where c.u. is channel use. The resulting curve is sketched in Figure 13.1–6.

To obtain such a curve for a certain E_b/N_0 or SNR_i on the channel, we first find p_b from Figure 13.1–5. Then we plot the corresponding vertical line on Figure 13.1–4,

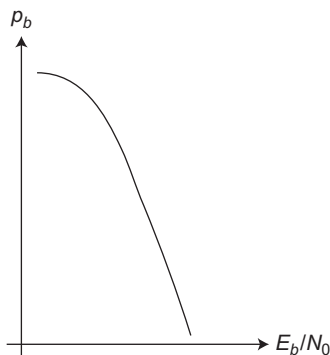


FIGURE 13.1–5

Probability of bit error versus channel SNR.

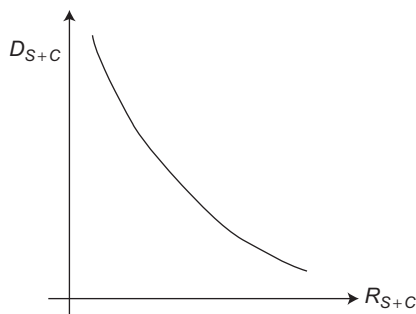


FIGURE 13.1–6

Total distortion versus source bits per channel use.

from which we can generate the plot in Figure 13.1–6 for a particular R_C of the used FEC, assumed modulation, and channel model. Some real applications to an ECSQ subband video coder and an MPEG 2 coder over an additive white Gaussian noise (AWGN) channel are contained in [22].

Error Concealment

Conventional source decoding does not make use of any *a priori* information about the video source, such as a source model or power spectral density, in effect making it an unbiased or maximum likelihood source decoder. However, better PSNR and visual performance can be expected by using model-based decoding, e.g., a post-processing filter. For block-based intraframe coders, early postprocessing concealed blocking artifacts at low bitrates by smoothing nearest neighbor columns and rows along DCT block boundaries.

Turning to channel errors and considering the typical losses that can occur in network transmission, various error-concealment measures have been introduced to counter the common situations where blocks and reference blocks in prior frames or motion vectors are missing, due to packet loss. If the motion vector for a block is missing, it is common to use an average of neighboring motion vectors to perform the needed prediction. If the coded residue or displaced frame difference (DFD) is missing for a predicted block (or region), often just the prediction is used without any update. If an *I* block (or region) is lost, then some kind of spatial prediction from the neighborhood is often used to fill in the gap. Such schemes can work quite well for missing data, especially in rather smooth regions of continuous motion. Directional interpolation based on local geometric structure has also been found useful for filling in missing blocks [23].

A particularly effective method to deal with lost or erroneous motion vectors was presented by Lam et al. [24]. A boundary-matching algorithm is proposed to estimate a missing motion vector from a set of candidates. The candidates include the motion vector for the same block in the previous frame, the available neighboring motion vectors, an average of available neighboring motion vectors, and the median of the available neighboring motion vectors. The estimate is then chosen that minimizes a side-match error—i.e., the sum of squares of the first-order difference across the available top, left, and bottom block boundaries. A summary of error concealment features in the H.26L test model is presented by Wang et al. [25].

Some error concealment features of H.264/AVC will be presented in Section 13.3. In connection with the standards, though, it should be noted that error concealment is *nonnormative*, meaning that error concealment is not a mandated part of the standard. Implementors are thus free to use any error concealment methods they find appropriate for the standard encoded bitstreams. When we speak of error concealment features of H.26L or H.264/AVC, we mean error concealment features of the test models for these coders. An overview of early error concealment methods is contained in the Katsaggelos and Galatsanos book [26].

13.2 ROBUST SWT VIDEO CODING (BAJIĆ)

In this section we introduce dispersive packetization, which can help the error concealment at the receiver to deal with losses. We also introduce a combination of MDC with FEC that is especially suitable for matching scalable and embedded coders to a best-efforts network such as the current Internet.

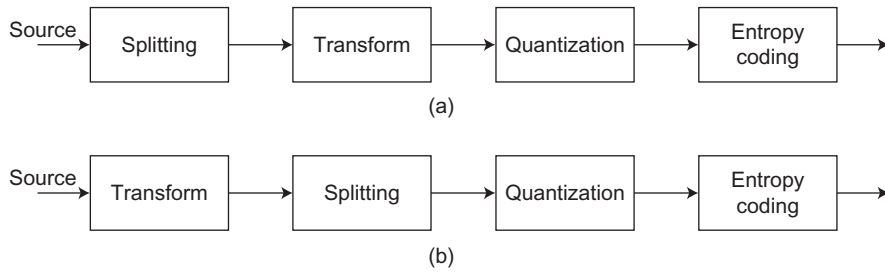
Dispersive Packetization

Error concealment is a popular strategy for reducing the effects of packet losses in image and video transmission. When some data from the compressed bitstream (such as a set of macroblocks, subband samples, and/or motion vectors) are lost, the decoder attempts to estimate the missing pieces of data from the available ones. The encoder can help improve estimation performance at the decoder side by packetizing the data in a manner that facilitates error concealment. One such approach is *dispersive packetization* (DP).

The earliest work in this area seems to be the even–odd splitting of coded speech samples by Jayant [27]. Neighboring speech samples are more correlated than samples that are far from each other. When two channels are available for speech transmission, sending even samples over one of the channels and odd samples over the other will facilitate error concealment at the decoder if one of the channels fails. This is because each missing even sample will be surrounded by two available odd samples, which can serve as a basis for interpolation, and vice versa. If multiple channels (or packets) are available, a good strategy would be to group together samples that are maximally separated from each other. For example, if N channels (packets) are available, the i th group would consist of samples with indices $\{i, i + N, i + 2N, \dots\}$. In this case, if any one group of samples is lost, each sample from that group would have $N - 1$ available neighboring samples on either side. Even–odd splitting is a special case of this strategy for $N = 2$. The idea of splitting neighboring samples into groups that are transmitted independently (over different channels, or in different packets), also forms the basis of DP of images and video.

DP of Images

In the 1-D example of a speech signal, splitting samples into N groups that contain maximally separated samples amounts to simple extraction of N possible phases of the speech signal subsampled by a factor of N . In the case of multidimensional signals, subsampling by a factor of N can be accomplished in many different ways (see Chapter 2 and also [28]). The particular subsampling pattern that maximizes the distance between the samples that are grouped together is the one that solves the sphere packing problem [29] in the signal domain. Digital images are usually defined on a subset of the 2-D integer lattice \mathbb{Z}^2 . A fast algorithm for sphere packing onto \mathbb{Z}^2 was proposed in [30], and it can be used directly for DP of raw images.

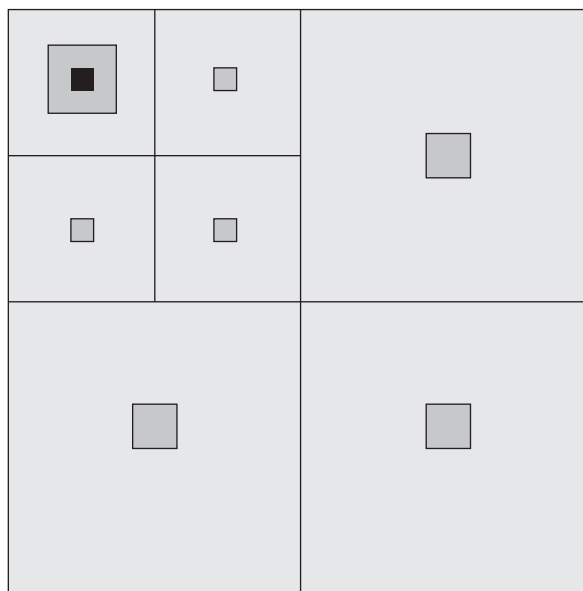
**FIGURE 13.2-1**

Splitting of source samples can be done (a) before the transform, or (b) after the transform.

In practice, however, we are often interested in transmitting transform-coded images. In this case, there are two ways to accomplish DP, as shown in Figure 13.2-1. Samples can be split either before (Figure 13.2-1a) or after the transform (Figure 13.2-1b). The former approach was used in [31], where the authors propose splitting the image into four subsampled versions and then encoding each version by the conventional JPEG coder. The latter approach was used in several works targeted at SWT-coded images [32, 33]. Based on the results reported in these papers, splitting samples before the transform yields lower compression efficiency but higher error resilience, while splitting after the transform gives higher compression efficiency with lower error resilience. For example, the results in [31] for the Lena image show the coding efficiency reduction of over 3 dB in comparison with conventional JPEG, while acceptable image quality is obtained with losses of up to 75%. On the other hand, the methods in [34] and [33] are only about 1 dB less efficient than SPIHT [35] (which is a couple of dB better than JPEG on the Lena image), but are targeted at lower losses, up to 10–20%.

We now describe the DP method of [33] in more detail. Figure 13.2-2 shows a two-level SWT of an image. One sample from the lowest frequency *LL* subband is shown in black, and its neighborhood in the space-frequency domain is shown in gray. In DP, we try to store neighboring samples in different packets. If the packet containing the sample shown in black is lost, many of its neighbors will still be available. Due to the intraband and interband relationships among subband/wavelet samples, the fact that many neighbors of a missing sample are available can facilitate error concealment.

Given the target image size and the maximum allowed packet size (both in bytes), we can determine the suitable number of packets, say N . Using the lattice partitioning method from [30], a suitable 2-D subsampling pattern $p: \mathbb{Z}^2 \rightarrow \{0, 1, \dots, N-1\}$ can be constructed for the subsampling factor of N . This pattern is used in the *LL* subband; the sample at location (i, j) in the *LL* subband is stored in the packet $p(i, j)$. Subsampling patterns for the higher frequency patterns are obtained by modulo shifting in the following way. Starting from the *LL* subband and going toward higher

**FIGURE 13.2-2**

One low-frequency SWT sample (black) and its space-frequency neighborhood (gray).

frequencies in a Z-scan, label the subbands in increasing order from 0 to K . Then, the sample at location (i, j) in subband k is stored in packet $(p(i, j) + k) \bmod N$. As an example, Figure 13.2-3 shows the resulting packetization into $N = 4$ packets of a 16×16 image with two levels of subband/wavelet decomposition. One LL sample from packet number 3 is shown in black. Note that out of a total of 23 space-frequency neighbors (shown in gray) of this LL sample, only three are stored in the same packet. For details on the entropy coding and quantization, refer to [33] and references therein.

As mentioned before, DP facilitates easy error concealment. Even simple error concealment algorithms can produce good results when coupled with DP. For example, in [33] error concealment is carried out in the three subbands (LL , HL , and LH) at the lowest decomposition level, where most of the signal energy is usually concentrated. Missing samples in the LL subband are interpolated bilinearly from the four nearest available samples in the horizontal and vertical directions. Missing samples in the LH and HL subbands are interpolated linearly from nearest available samples in the direction in which the subband has been lowpass filtered. Missing samples in higher frequency subbands are set to zero. More sophisticated error concealment strategies may bring further improvements [36]. As an example, Figure 13.2-4 shows PSNR versus packet loss comparison between the *packetized zerotree wavelet* (PZW) method from [34] and two versions of DP—one paired with bilinear concealment [32] and the other with adaptive maximum *a posteriori*

0	1	2	3	1	2	3	0	0	1	2	3	0	1	2	3
2	3	0	1	3	0	1	2	2	3	0	1	2	3	0	1
0	1	2	3	1	2	3	0	0	1	2	3	0	1	2	3
2	3	0	1	3	0	1	2	2	3	0	1	2	3	0	1
2	3	0	1	3	0	1	2	0	1	2	3	0	1	2	3
0	1	2	3	1	2	3	0	2	3	0	1	2	3	0	1
2	3	0	1	3	0	1	2	0	1	2	3	0	1	2	3
0	1	2	3	1	2	3	0	2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3
1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3
1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3
1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3
1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3

FIGURE 13.2-3

Example of a 16×16 image with two levels of SWT decomposition, packetized into four packets.

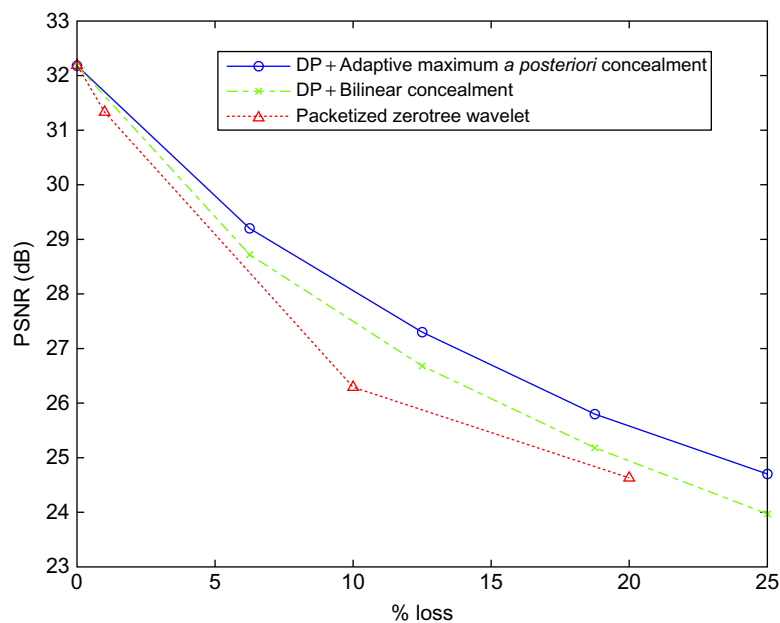


FIGURE 13.2-4

PSNR versus packet loss on the 512×512 monochrome Lena image.

concealment [36]. The example is for the 512×512 grayscale Lena image encoded at 0.21 bpp.

DP of Video

Two-dimensional concepts from the previous section can be extended to three dimensions for DP of video. For example, Figure 13.2–5 shows how alternating the packetization pattern from frame to frame (by adding 1 modulo N) will ensure that samples from a common SWT neighborhood will be stored in different packets. This type of packetization can be used for intraframe-coded video. When motion compensation is employed, each level of the motion-compensated spatiotemporal pyramid consists of an alternating sequence of frames and motion vector fields. In this case, we want to ensure that motion vectors are not stored in the same packet as the samples they point to. Again, adding 1 modulo N to the packetization pattern used for samples will shuffle the pattern so that it can be used for motion vector packetization, as shown in Figure 13.2–6.

To see the performance of video DP, we present an example that involves the grayscale SIF *football* sequence at 30 fps. The sequence was encoded at 1.34 Mbps with a group of pictures (GOP) size of four frames using the DP video coder from Bajic and Woods [33]. The Gilbert model was used to simulate transmission over a

	2	3	0	1	3	0	1	2	2	3	0	1	2	3	0	1
	0	1	2	3	1	2	3	0	0	1	2	3	0	1	2	3
	2	3	0	1	2	3	1	2	3	0	0	1	2	3	0	1
	0	1	2	3	1	2	3	0	2	3	0	1	2	3	0	1
	0	1	2	3	1	2	3	0	0	1	2	3	0	1	2	3
	2	3	0	1	3	0	1	2	2	3	0	1	2	3	0	1
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	2	3	0	1	3	0	1	2	0	1	2	3	0	1	2	3
	0	1	2	3	1	2	3	0	2	3	0	1	2	3	0	1
	2	3	0	1	3	0	1	2	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	0	1	2	3	0	1	2	3
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	2	3	0	1	2	3	0	1
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	2	3	0	1	2	3	0	1
	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2
	1	2	3	0	1	2	3	0	0	1	2	3	0	1	2	3
	3	0	1	2	3	0	1									

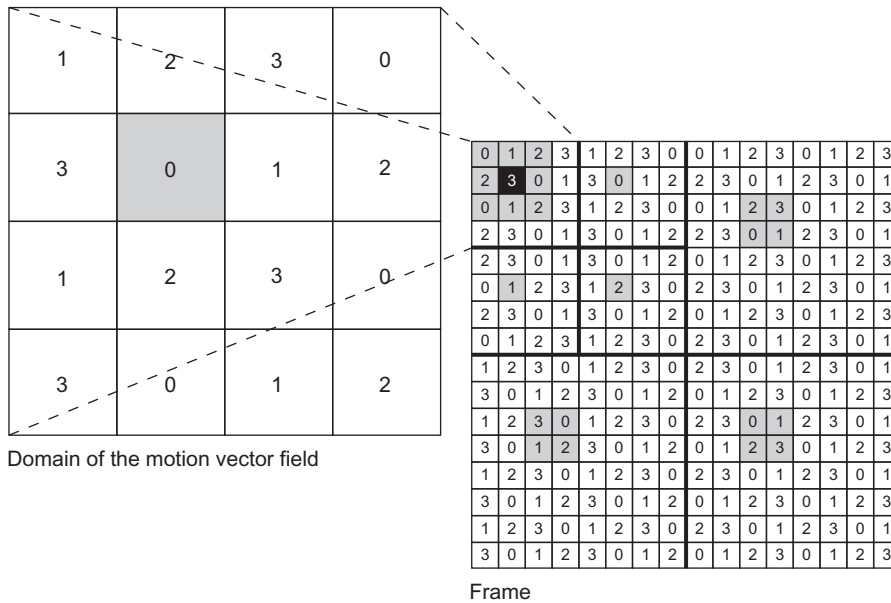


FIGURE 13.2-6

DP of motion-compensated SWT video.

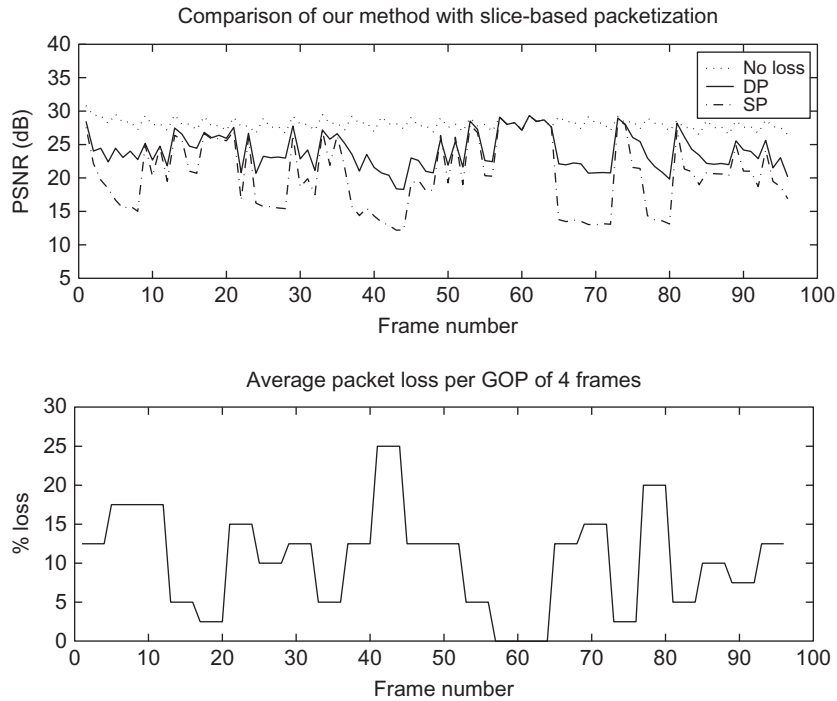
packet-based network, with the “good” state representing packet reception and the “bad” state representing packet loss. Average packet loss was 10.4%.

Figure 13.2-7 shows the results of a sample run from the simulation. The top part shows the PSNR of the decoded video in three cases: no loss, DP, and *slice-based packetization* (SP). The SP video is similar to the H.26x and MPEGx recommendations for packet-lossy environments, where each slice (a row of macroblocks) is stored in a separate packet. The bottom part of the figure shows the corresponding packet loss profile. In this case, DP provides an average 3- to 4-dB advantage in PSNR over SP.

Visual comparison between DP and SP is shown in Figure 13.2-8, which corresponds to frame 87 of the football sequence, with 10% loss. The figure shows (a) original frame, (b) coded frame with no loss (PSNR = 27.7 dB), (c) decoded SP frame (PSNR = 20.6 dB), and (d) decoded DP frame (PSNR = 22.2 dB); e and f are zoomed-in segments of c and d, respectively, illustrating the performance of the two methods on small details. Observe that details are better preserved in the DP frame. Video is available for download at the book’s Web site.

Multiple Description FEC

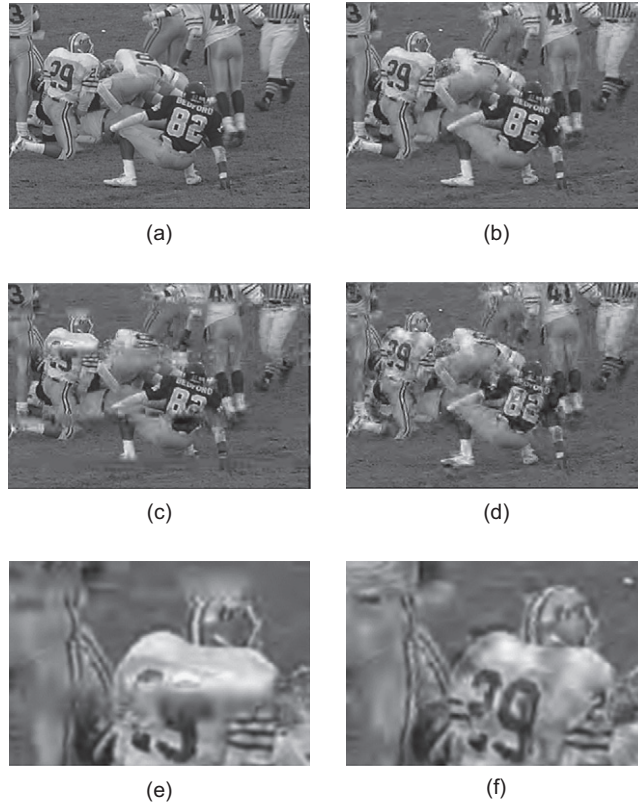
The *multiple description forward error correction* (MD-FEC) technique is a way of combining information and parity bits into descriptions or packets in order to achieve robustness against packet loss through unequal error protection. Among the

**FIGURE 13.2-7**

Comparative performance of DP and SP schemes on the football sequence.

first works on the topic was the *priority encoding transmission* (PET) of Albanese et al. [37]. In PET, a message is broken into smaller segments, and each segment is assigned a priority level between 0 and 1. Then, each segment is encoded into a set of packets using a version of the RS error-control codes in a way that guarantees that the segment is decodable if a fraction of the number of packets at least equal to its priority, is received. Later, MD-FEC [38, 39] used a very similar strategy, but assumed that the message (image or video clip) is encoded into an embedded or SNR scalable bitstream. This also facilitates assignment of priorities based on distortion-rate ($D-R$) characteristics. Our description of the method follows the work of Puri et al. [38].

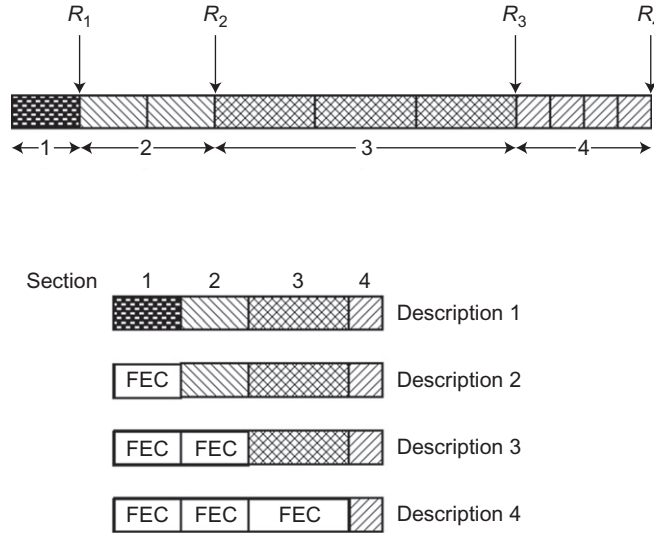
Consider a message encoded into a scalable and embedded bitstream. The bitstream is first divided into N sections: $(0, R_1], (R_1, R_2], \dots, (R_{N-1}, R_N]$. Section k is split into k subsections, and encoded by a (N, k) RS code. Such a code can correct up to $N - k$ erasures. Individual symbols of RS codewords are concatenated to make descriptions. Each description is stored in a separate packet, so we will use the terms “description” and “packet” interchangeably. An illustration of the MD-FEC procedure for $N = 4$ is given in Figure 13.2-9. In this example, section 1 is protected

**FIGURE 13.2-8**

DP demonstration: frame 87 of the football sequence; SIF at 1.34 Mbps.

by RS (4,1) code, section 2 by RS (4,2) code, section 3 by RS (4,3) code, while section 4 is unprotected. If no packets are lost, the bitstream can be decoded up to R_4 . If one packet is lost, we are guaranteed to be able to decode up to R_3 , since RS (4,1) can correct any single erasure. However, if the one packet that was lost was packet 4, then we can decode all the way up to $R_3 + 3(R_4 - R_3)/4$. In general, if j out of N packets are received, we can decode up to some rate in the range $[R_j, R_j + j(R_{j+1} - R_j)/(j + 1)]$.

Given the channel packet-loss characteristics, the goal of FEC assignment in MD-FEC is to minimize the expected distortion subject to a total rate constraint. In an embedded bitstream, we have $D(R_i) \leq D(R_j)$ for $R_i > R_j$, and this fact can be used to reduce computational burden. Designing the MD-FEC amounts to a joint optimization over both the source coder (i.e., how much of the source bitstream to take) and channel coder (i.e., how much parity [FEC chunks] to insert).

**FIGURE 13.2-9**

A simple illustration of MD-FEC.

If we characterize the channel as a packet-loss channel, this means that packets can be lost but not corrupted. Letting the probability of exactly k packets being lost be p_k , we can write the overall expected distortion as

$$D_{\text{avg}} = \sum_{k=1}^N p_k D(R_k) \quad (13.2-1)$$

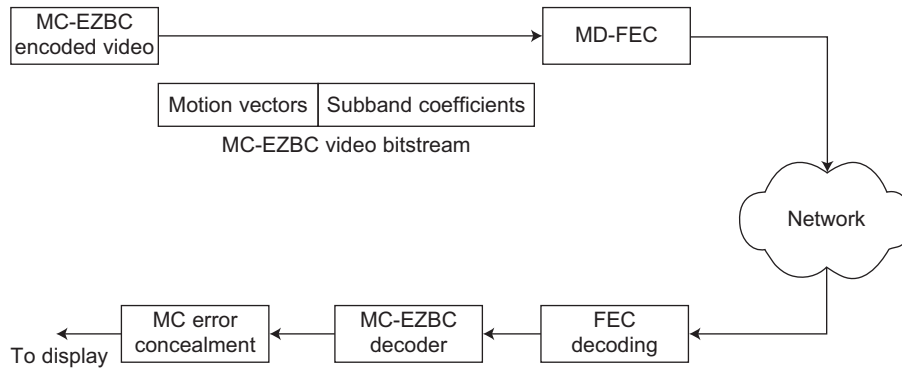
and seek to minimize this quantity over all choices of source-rate breakpoints $0 \leq R_1 \leq R_2 \leq \dots \leq R_N$ such that the bandwidth B of the channel is not exceeded. We can write this channel bandwidth constraint as

$$\sum_{k=1}^N \alpha_k R_k \leq B, \quad (13.2-2)$$

for an appropriate choice of the constants α_k (see Figure 13.2-9 and end-of-chapter problem 4). For a convex $D(R)$ function, we have that D_{avg} is a convex function of rate vector (R_1, R_2, \dots, R_N) subject to inequality constraints and can be solved by any number of optimization methods. The reader is referred to [35] for a possible fast solution.

Robust 3D-SPIHT

The 3D-SPIHT spatiotemporal extension [40] of the SPIHT image coder (cf. Section 9.6), was used in [38] as the video coder to produce an embedded bitstream for MD-FEC. In a typical application, MD-FEC would be applied on a GOP by GOP

**FIGURE 13.2–10**

Video streaming system based on MC-EZBC.

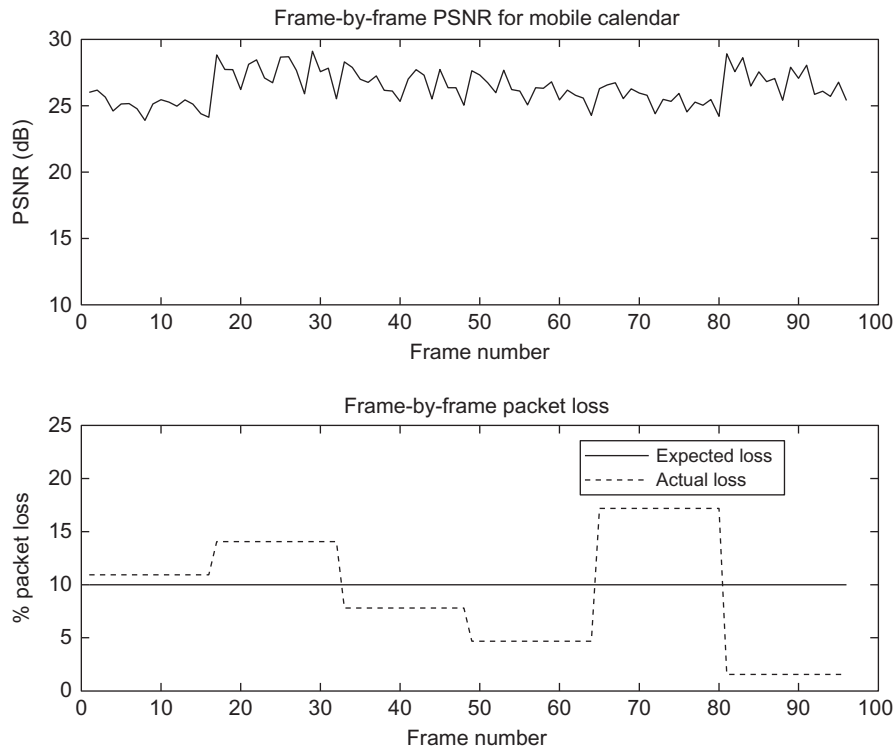
basis. Channel conditions are monitored throughout the streaming session, and for each GOP, an FEC assignment is carried out using recent estimates of the channel loss statistics.

A Robust MC-EZBC

One can also use MC-EZBC (cf. Section 12.5) to produce an embedded bitstream for MD-FEC. Since motion vectors form the most important part of interframe video data, they would be placed first in the bitstream, followed by SWT samples encoded in an embedded manner. A video streaming system based on MC-EZBC was proposed in [41]. The block diagram of the system is shown in Figure 13.2–10. As before, MD-FEC is applied on a GOP-by-GOP basis. A major difference from the system based on 3D-SPIHT [40] is the MC error-concealment module at the decoder. Since motion vectors receive the most protection; they are the most likely part of the bitstream to be received. MC error concealment is used to recover from short burst losses that are hard to predict by simply monitoring channel conditions and are therefore often not compensated for by MD-FEC.

Example 13.2–1: MD-FEC

This example illustrates the performance of MD-FEC for video transmission. The Mobile Calendar sequence (SIF resolution, 30 fps) was encoded into an SNR-scalable bitstream using the MC-EZBC video coder. The sequence was encoded with a GOP size of 16 frames. FEC assignment was computed for each GOP separately, with 64 packets per GOP and a total rate of 1.0 Mbps, assuming a random packet loss of 10%. Video transmission was then simulated over a two-state Markov or Gilbert channel, which is often used to capture the bursty nature of packet loss in the Internet. This channel has two states: “good,” corresponding to successful packet reception, and “bad,” corresponding to packet loss. The channel was simulated for 10% average loss rate and an average burst length of two packets.

**FIGURE 13.2-11**

MD-FEC simulation result on the mobile calendar clip.

Results from a sample simulation run are shown in [Figure 13.2-11](#). The top part of the figure shows PSNR for 96 frames of the sequence. The bottom part shows the expected loss rate of 10% (solid line) and the actual loss (dashed line). As long as the actual loss does not deviate much from the expected value used in FEC assignment, MD-FEC is able to provide consistent video quality at the receiver. (See robust SWT video results at this book's Web site.)

13.3 ERROR-RESILIENCE FEATURES OF H.264/AVC

The current VCEG/MPEG coding standard H.264/AVC inherits several error-resilient features from earlier standards such as H.263 and H.26L, but also adds several new ones. These features of the source coder are intended to work together with the transport error protection features of the network to achieve a needed QoS. The error-resilience features in H.264/AVC are syntax, data partitioning, slice interleaving, flexible macroblock ordering, SP/SI switching frames, reference frame selection, intrablock refreshing, and error concealment [42].

H.264/AVC is conceptually separated into a *video coding layer* (VCL) and a *network abstraction layer* (NAL). The output of the VCL is a slice, consisting of an integer number of macroblocks. These slices are independently decodable due to the fact that positional information is included in the header, and no spatial references are allowed outside the slice. Clearly, the shorter the slice, the lower the probability of loss or other compromise. For highest efficiency, there should be one slice per frame, but in poorer channels or loss environments, slices consisting of even a single row of macroblocks have been found useful. The role of the NAL is to map slices to transmission units of the various types of networks, e.g., map to packets for an IP-based network.

Syntax

The first line of defense from channel errors in video coding standards is the semantics and syntax [42]. If illegal syntax elements are received, then it is known that an error has occurred. At this point, error correction can be attempted or the data declared lost, and error concealment can commence.

Data Partitioning

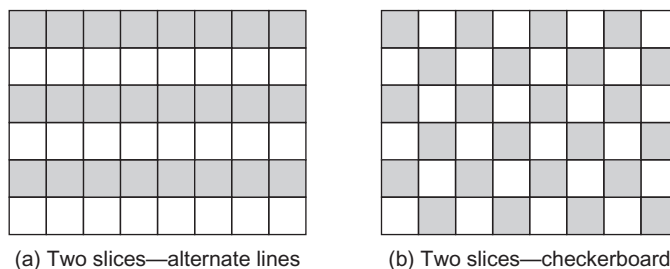
By using the *data partitioning* (DP³) feature, more important data can be protected with stronger error-control features, such as FEC and acknowledgement-retransmission, to realize unequal error protection (UEP). In H.264/AVC, coded data may be partitioned into three partitions: A, B, and C. In partition A is header information including slice position. Partition B contains coded block patterns for intra blocks, followed by their transform coefficients. Partition C contains coded block patterns for inter blocks and their transform coefficients. Coded block patterns contain the position and size information for each type of block, and incur a small overhead penalty for choosing DP.

Slice Interleaving and Flexible Macroblock Ordering

An innovative feature of H.264/AVC is *flexible macroblock ordering* (FMO) that generalizes slice interleaving from earlier standards. The FMO option permits macroblocks to be distributed across slices in such a way as to aid error concealment when a single slice is lost. For example, we can have two slices, as indicated in Figure 13.3–1 (a or b). The indicated squares can be as small as macroblocks of 16×16 pixels, or they could be larger. Figure 13.3–1a shows two interleaved slices, the gray and the white. If one of the two interleaved slices is lost, in many cases it will be possible to interpolate the missing areas from the remaining slice, both above and below.

In Figure 13.3–1b, a gray macroblock can more easily be interpolated if its slice is lost because it has four nearest neighbor macroblocks from the white slice, presumed received. On the other hand, such extreme macroblock dispersion as shown in this

³Please do not confuse with dispersive packetization. The meaning of “DP” should be clear in context.

**FIGURE 13.3-1**

An interleaved and FMO mapping of macroblocks onto two slices: the gray and the white slice.

checkerboard pattern would mean that the nearest neighbor macroblocks could not be used in compression coding of a given macroblock, resulting in loss of compression efficiency. Of course, we must trade off the benefits and penalties of an FMO pattern for a given channel error environment. One can think of FMO as a type of dispersive packetization.

Switching Frames

This feature allows switching between two H.264/AVC bitstreams stored on a video server. They could correspond to different bitrates, in which case the switching permits adjusting the bitrate at the location of the so-called *SP* or switching frame, while not waiting for an *I* frame. While not as efficient as a *P* frame, the *SP* frame can be much more efficient than an *I* frame. Another use for a switching frame is to recover from a lost or corrupted frame, which otherwise would propagate error until reset by an *I* frame or slice. This can be accomplished, given the existence of such switching frames, by storing two coded bitstreams in the video server, one with a conventional reference frame choice of the immediate past frame, and the second bitstream with a reference frame choice of a frame further back, say 2 or 3 frames back. Then if the receiver detects the loss of a frame, and this loss can be fed back to the server quickly enough, the switch can be made to the second bitstream in time for the computation and display of the next frame, possibly with a slight freeze-frame delay. The *SP* frame can also be used for fast-forward searching through the video program.

With reference to [Figure 13.3-2](#), we illustrate a switching situation, where we want to switch from the bottom bitstream 1 to the top bitstream 2. Normally this could only be done at the time of an *I* frame, since the *P* frame from bitstream 2 needs the decoded reference frame from bitstream 2 in order to decode properly. If the decoded frame from bitstream 1 is used, as in our switching scenario, then the reference would not be as expected and an error would occur and propagate via intra prediction within the frame and via inter prediction to the following frames. Note that this can just as well be done on a slice basis as well as the frame basis we are considering here.

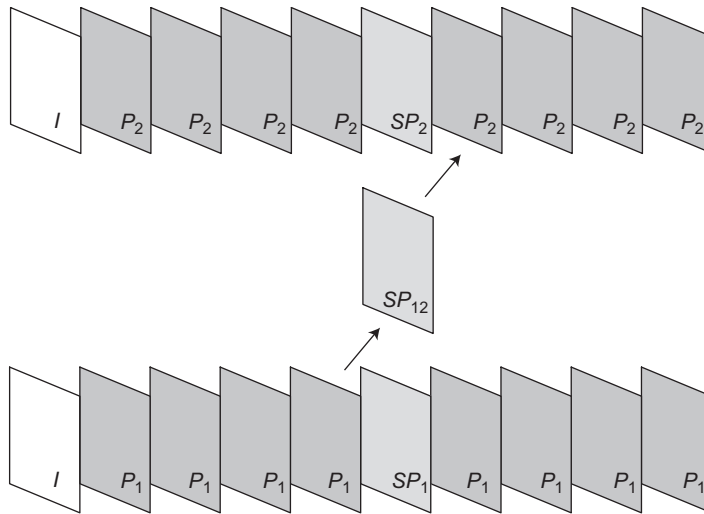
**FIGURE 13.3–2**

Illustration of switching from bottom bitstream 1 to top bitstream 2 via switching frame SP_{12} .

The question now is how can this switching be done in such a way as to avoid error in the frames following the switch. First, the method involves special *primary* switching frames SP_1 and SP_2 inserted into the two bitstreams at the point of the expected switch. The so-called *secondary* switching frame SP_{12} then must have the same reconstructed value as SP_2 , even though they have different reference frames and hence different predictions. The key to accomplishing this is to operate with the quantized coefficients in the block transform domain, instead of making the prediction in the spatial domain. The details are shown by Karczewicz and Kurceren [43], where a small efficiency penalty is noted.

Reference Frame Selection

In H.264/AVC we can use frames for reference other than the immediately last one. If the video is being coded now, rather than having been coded previously and stored on a server, then feedback information from the receiver can indicate when one of the reference frames is lost; thus the *reference frame selection* (RFS) feature can alter the choice of reference frames for encoding the current frame, in such a way as to exclude the lost frame, and therefore terminate any propagating error. Note that so long as the feedback arrives within a couple of frame times, it can still be useful in terminating propagation of an error in the output, which would normally only end at an *I* frame or perhaps at the end of the current slice. So RFS can make use of the multiple reference frame capability of H.264/AVC. Thus multiple reference frames also have a role in increasing the error resilience.

Intrablock Refreshing

A basic error-resilience method is to insert intraslices instead of intraframes. These intra-slices can be inserted in a random manner, so as not to increase the local data rate the way that intraframes do. This has been found to be desirable for keeping delay down in visual conferencing. Intraslices terminate error propagation within the slice, the same way that *I* frames do for the whole frame.

Error Concealment in H.264/AVC

While error concealment is not part of the normative H.264/AVC standard, there are certain nonnormative (informative) features in the test model [15]. First, a lost macroblock can be concealed at the decoder using a similar type of directional interpolation (prediction) as used in the coding loop at the encoder. Second, an interframe error concealment can be made from the received reference blocks, wherein the candidate with the smallest boundary matching error [24, 25] is selected, with the zero motion block from the previous frame always a candidate.

Example 13.3–1: Selection of Slice Size (Soyak and Katsaggelos)

In this example, baseline H.264/AVC is used to code slice sizes consisting of both one row and three rows of macroblocks. The foreman CIF clip was used at 30 fps and 384 Kbps. Because of the increased permitted use of intra prediction by the larger slice size, the average luma PSNRs without any packet loss were 36.2 and 36.9 dB, respectively. Then the packets were subjected to a loss simulation as follows. The smaller packets from the one-row slices were subjected to a 1% packet loss rate, while the approximately three-times longer packets from the three-row slices were subjected to a comparable 2.9% packet loss rate. Simple error concealment was used to estimate any missing macroblocks using a median of neighboring motion vectors.

The resulting average luma PSNRs became 30.8 and 25.6 dB, respectively. Typical frames grabbed from the video are shown in Figures 13.3–3 and 13.3–4. We can see



FIGURE 13.3–3

This slice consists of one row of macroblocks.

**FIGURE 13.3–4**

This slice consists of three contiguous rows of macroblocks.

the greater effect of data loss in [Figure 13.3–4](#). Video results are available in the Network Video folder at this book’s Web site. ■■■

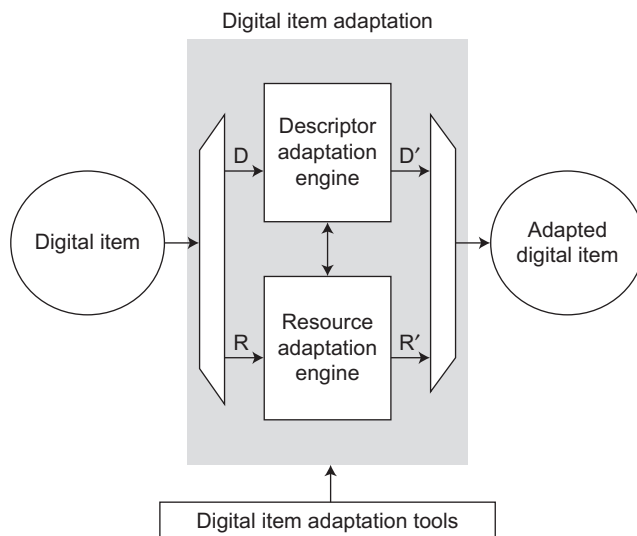
13.4 JOINT SOURCE–NETWORK CODING

Here, we talk about methods for network video coding that make use of *overlay networks*. These are networks that exist on top of the public Internet at the application level. We will use them for transcoding or data item adaptation and in an extension to MD-FEC. By *joint source–network coding* (JSNC), we mean that the source coding and network coding are performed jointly in an effort to optimize the performance of the combined system. Data item adaptation can be considered as an example of JSNC. Finally, we provide an introduction to the capacity-achieving improvement over routing that is called *network coding*, focusing on practical network coding of multiple description packets.

Digital Item Adaptation in MPEG 21

MPEG 21 has the goal of making media, including video, available everywhere [44, 45], and this includes heterogeneous networks and display environments. Hence there is a need for adaptation of these digital items in quality (bitrate), spatial resolution, and frame rate within the network. The digital item adaptation (DIA) engine, as shown in [Figure 13.4–1](#), was proposed to accomplish this.

Digital items such as video packets enter on the left and are operated upon by the resource adaptation engine for transcoding to the output-modified digital item. At the same time, the descriptor field of this digital item needs to be adapted also, to reflect the modified or remaining content properly, e.g., if 4CIF is converted to CIF, descriptors of the 4CIF content would be removed with only the CIF content descriptors remaining. There would also be backward flows from the receivers, indicating the needs of the downstream users as well as information about network congestion,

**FIGURE 13.4-1**

A concept DIA model from ISO document [45].

available bitrates, and bit-error rates, that the DIA engine must consider also in the required adaptation. A general overview of video adaptation was given by Chang and Vetro [46].

Fine Grain Adaptive FEC

We have seen that JSNC uses an overlay infrastructure to assist video streaming to multiple users simultaneously by providing lightweight support at intermediate overlay nodes. For example, in Figure 13.4-2, overlay *data service nodes* (DSNs) construct an overlay network to serve heterogeneous users. Users *A* to *G* have different video requirements (shown as “frame rate/resolution/available bandwidth”), and P_a to P_g are the packet-loss rates of different overlay virtual links. We do not consider the design of the overlay network here. While streaming, the server sends out a single *fine-grain-adaptive-FEC* (FGA-FEC) coded bitstream based on the highest user requirement (in this case 30/CIF/3M) and actual network conditions. FGA-FEC divides each network packet into small blocks and packetizes the FEC-coded bitstream in such a way that if any original data packets are actively dropped (adapted by the DIA engine), the corresponding information in parity bits is also completely removed. The intermediate DSNs can adapt the FEC-coded bitstream by simply dropping a packet or shortening a packet by removing some of its blocks. Since there is no FEC decoding/re-encoding, JSNC is very efficient in terms of computation. Furthermore, the data manipulation is at block level, which is precise in terms of adaptation.

In this section, we use a fully scalable MC-EZBC video coder [47] to show the FGA-FEC capabilities. As seen in Chapter 12, MC-EZBC produces embedded

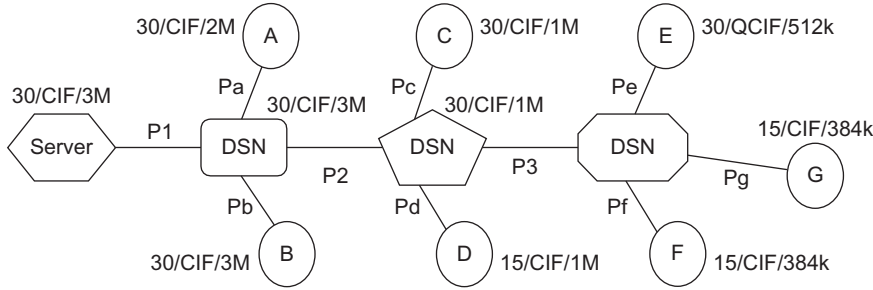


FIGURE 13.4-2

An example overlay network.

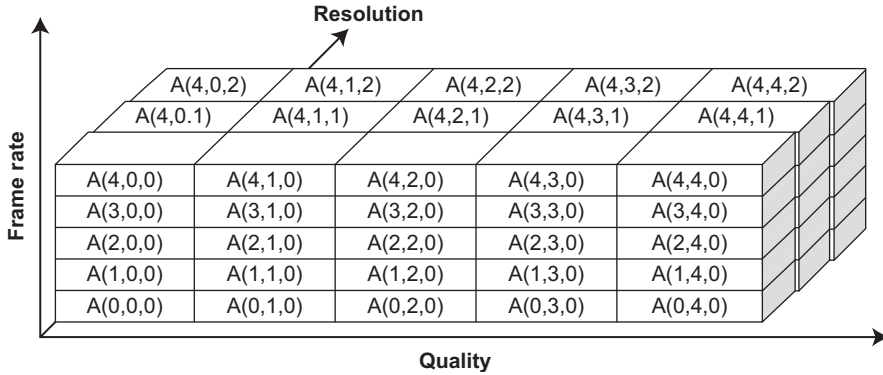


FIGURE 13.4-3

Atom diagram of video scalability dimensions [48].

bitstreams supporting three types of scalabilities—temporal, spatial, and SNR. Here, we use the same notation as [47]. Each GOP coding unit consists of independently decodable bitstreams $\{Q^{MV}, Q^{YUV}\}$. Let $l_t \in \{1, 2, \dots, L_t\}$ denote the temporal scale; then the MV bitstream Q^{MV} can be divided into temporal scales and consists of $Q_{l_t}^{MV}$ for $2 \leq l_t \leq L_t$. Let $l_s \in \{1, 2, \dots, L_s\}$ denote the spatial scales. The subband coefficient bitstream Q^{YUV} is also divided into temporal scales and further divided into spatial scales as Q_{l_t, l_s}^{YUV} , for $2 \leq l_t \leq L_t$ and $1 \leq l_s \leq L_s$. For example, the video at one-quarter spatial resolution⁴ and one-half frame rate is obtained from the bitstream as $Q = \{Q_{l_t}^{MV} : 1 \leq l_t \leq L_t - 1\} \cup \{Q_{l_t, l_s}^{YUV} : 1 \leq l_s \leq L_s - 1\}$. In every sub-bitstream Q_{l_t, l_s}^{YUV} , the luma and chroma subbands Y , U , and V are progressively encoded from the most significant bitplane to the least significant bitplane (cf. Fully Embedded SWT Coders in Section 9.6).

⁴By ‘one-quarter spatial resolution,’ we mean a factor of one-half in each spatial dimension.

Scaling in terms of quality is obtained by stopping the decoding process at any point in bitstream Q . The MC-EZBC-encoded bitstream can be further illustrated as *digital items*, as in Figure 13.4-3 [48], which shows the video bitstream in view of three forms of scalability. The video bitstream is represented in terms of *atoms*, which are fractional bitplanes [47]. The notation $A(F, Q, R)$ represents an atom of (frame-rate, quality, resolution) scalability. Choosing a particular casual subset of atoms corresponds to scaling the resulting video to the desired resolution, frame rate, and quality. These small pieces of bitstream are interlaced in the embedded bitstream. Intermediate DSNs adapt the digital items according to user preferences and network conditions. Since the adaptation can be implemented as simple dropping of corresponding atoms, DSNs do not need to decode and re-encode the bitstream, making them very efficient. On the other hand, the adaptation is done based on atoms in a bitstream, which can approximate the quality of pure source coding.

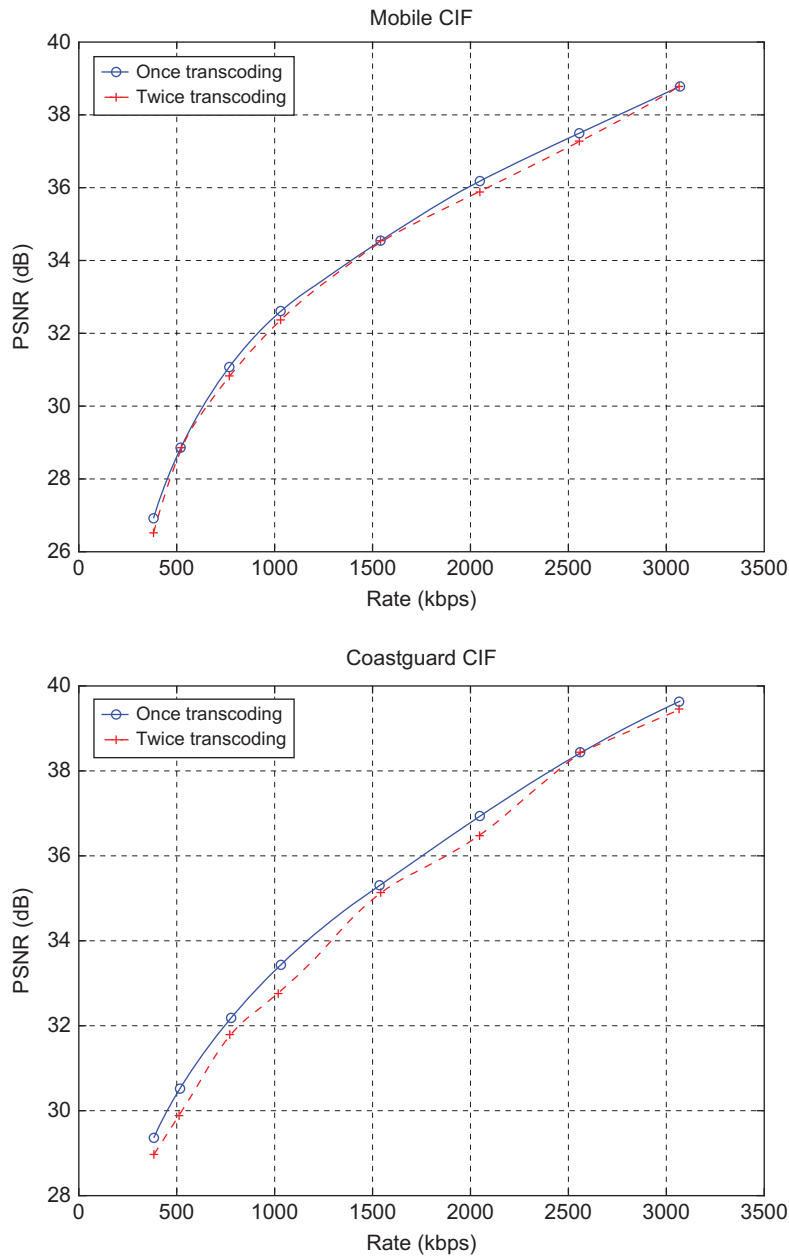
Example 13.4-1: Adaptation Example

Using the MC-EZBC fully scalable video coder, we have adapted the bitrate in the network by including limited overhead information in the bitstream, allowing bitrate adaptation across a range of bitrates for the coastguard and mobile calendar CIF test clips. Figure 13.4-4 shows the resulting PSNR plot versus bitrate. Two plots are given, corresponding to the first extraction at the server, and then a secondary extraction, corresponding to DIA within the network using six quality layers, whose starting locations are carried in a small header. We can see that the PSNR performance is only slightly reduced from that at the video server, where full scaling information is available.

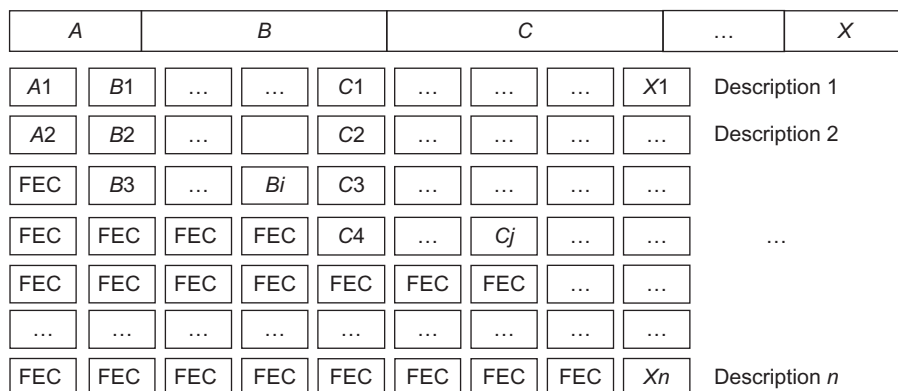
Digital Item Adaptation can be done for nonscalable formats too, but only via the much more difficult method of transcoding. Next, we show a method of combining DIA with FEC to increase robustness.

DSNs adapt the video bitstream based on user requirements and available bandwidth. When parts of the video bitstream are actively dropped by the DIA engine, FEC codes need to be updated accordingly. This update of FEC codes has the same basic requirements as the video coding—efficiency (low computation cost) and precision (if a part of the video data is actively dropped, parity bits protecting that piece of data should also be removed). Based on these considerations, a precise and efficient FGA-FEC scheme has been proposed [49, 50] based on RS codes.

Given a piece of video bitstream, shown in Figure 13.4-5 (top line), divided into sections as A, B, C, \dots, X , the FGA-FEC further divides each section of bitstream into equal-size blocks. Smaller block size means finer granularity and better adaptation precision. In the lower part of Figure 13.4-5, the bitstream is applied vertically onto the blocks as $(A1, A2; B1, \dots, Bi; C1, \dots, Cj; \dots; X1, \dots, Xn)$. The RS coding computation is also applied vertically across these blocks to generate the parity blocks,

**FIGURE 13.4-4**

PSNR performance comparison (Mobile and Coastguard test clips in CIF format) versus bitrate for DIA operation in network.

**FIGURE 13.4-5**

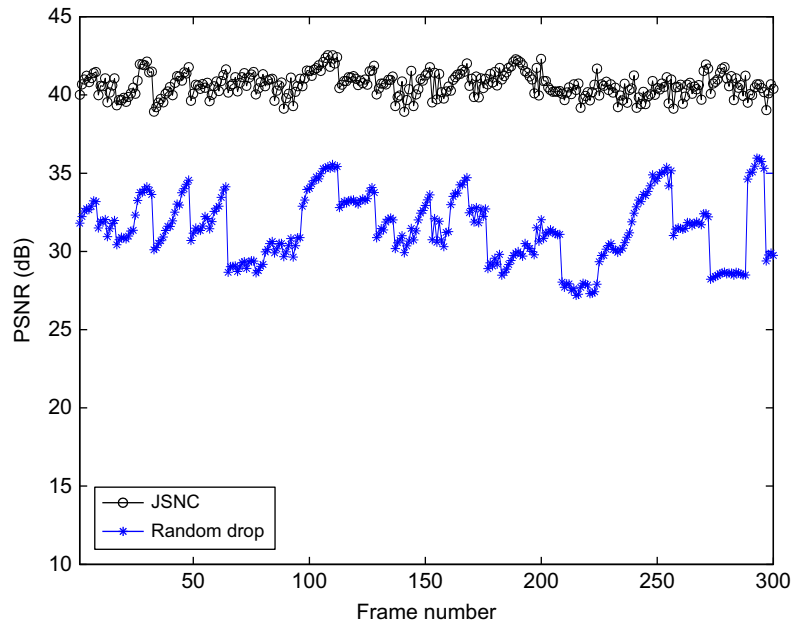
FGA-FEC coding scheme.

marked “FEC” in the figure. Because both the source data mapping and the FEC are performed vertically, and at a fine-grain (small) block size, FGA-FEC can be much more efficient for DIA than can MD-FEC, where the data was mapped horizontally onto the relatively coarse sections. The optimal allocation of FEC to different sections was described in [Section 13.2](#) and in [14, 41]. After FEC encoding, each horizontal row of blocks is packetized as one description, i.e., one description is equivalent to one network packet.

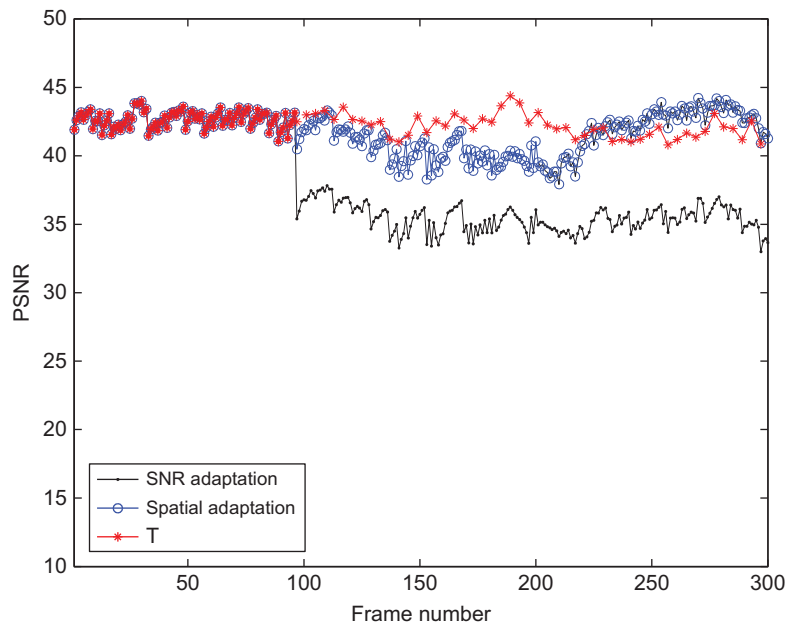
Similar to MD-FEC in [Section 13.2](#), FGA-FEC transforms a priority embedded bitstream into nonprioritized descriptions to match a best-efforts network. In addition, the FGA-FEC scheme has the ability of fine granular adaptation at block level. In our experiments, we set the block level to one byte, which matches the RS code symbol size. To facilitate intermediate overlay node adaptation, an information packet is sent ahead of one GOP to tell the intermediate nodes about the block size, FEC codes, and bitstream information at the block level.

Example 13.4-2: Overlay Network Adaptation

Traditionally, when network congestion occurs, data packets are randomly dropped to avoid congestion. To avoid this, the JSNC scheme adapts the packets in the intermediate network nodes to reduce the bandwidth requirement. Given a 1.5-Mbps bitstream and an available bandwidth of 1455 Kbps, in [Figure 13.4-6](#) we compare PSNR-Y of JSNC to a random packet-drop scheme with a 3% packet-loss rate. Observe that the proposed scheme significantly outperforms random drop by about 10 dB. In [Figure 13.4-7](#) we show objective video quality (PSNR) when the available bandwidth changes. Originally, the user is receiving a 2-Mbps, CIF format, 30-fps bitstream, but starting with frame 100, the user has only 512 Kbps.

**FIGURE 13.4-6**

JSNC versus random packet drop.

**FIGURE 13.4-7**

3-D adaptation in frame rate, resolution, and SNR (bitrate).

There are three possible choices for the user: (1) SNR adaptation to 512 Kbps, (2) temporal adaptation to one-quarter of the original frame rate, or (3) spatial adaptation down to QCIF resolution. Options 2 and 3 need additional SNR adaptation to 512 Kbps. With FGA-FEC, the users can choose their preference based on their application needs. We note from Figure 13.4–7, however, that there is a significant PSNR penalty to pay (about 7 dB) for staying with the full frame rate and spatial resolution, and just cutting down on the bitrate. See the Network Video folder at this book's Web site for video results. ■

Network Coding of Video

In unicast routing, information-bearing packets arrive at network routers, which look only at the packet headers and forward them on an appropriate outgoing link. In the case of video multicast, these packets generally must be replicated and sent out on more than one outgoing link to their multiple destinations. This process is termed *multicast routing* and is much more efficient than unicast for simultaneously sending a given video to many receivers. However, it turns out that there is a more efficient method called *network coding* that was first published by Ahlswede et al. [51] in 2000. Network coding is a generalization of multicast routing (routing with replication) wherein the outgoing packets are additionally allowed to be linear combinations of incoming packets, the exact linear combination being the network code. The original example shown to justify the claim for improved performance is the *butterfly network* shown in Figure 13.4–8, consisting of one source S and two receivers R_1 and R_2 connected by nine directed links and four intermediate nodes. This is an idealized graph model of a network, consisting of edges and nodes, and each edge has a capacity of 1 bit per use (bpu).

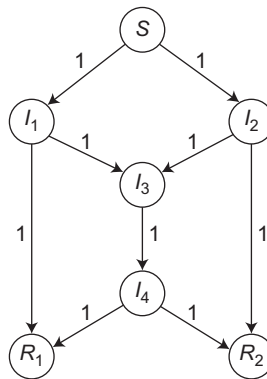
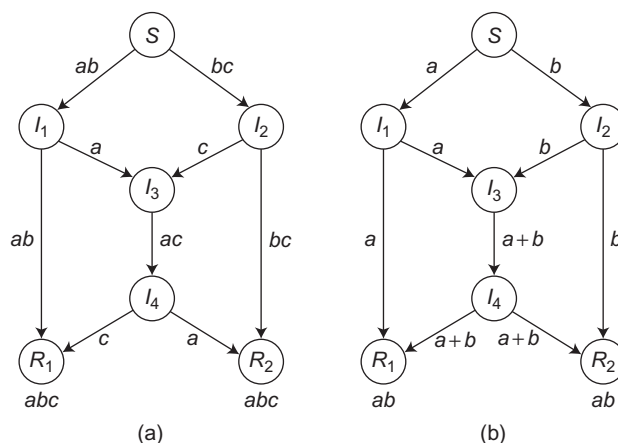


FIGURE 13.4–8

Simple directed graph network called *butterfly*.

**FIGURE 13.4-9**

Butterfly net: (a) **routing**, showing two uses of each link and with replication, (b) **network coding**, showing one use of each link.

Figure 13.4-9a shows multicast routing for the problem of sending three binary messages, a , b , and c , from the source to both receivers in two network (graph) uses. Thus the throughput of multicast routing is 1.5 bpu. Figure 13.4-9b shows the network coding solution, wherein the central node I_3 outputs the sum of messages a and b , and this enables the two receivers to get two messages a and b in one network use, for a throughput of 2 bpu. It is also shown in [51] that this is actually the capacity of the butterfly network in the sense that no coding scheme can do better. They show that the *multicast capacity* of a network with one source and multiple receivers can be found as the max-flow of the min-cut. In other words, if you cut the network graph into two pieces, one containing the source and the other containing a receiver, and then record the net flow across the boundary from the source to the receiver, then the minimum of these max flows across these cuts is the max-flow min-cut capacity. Li et al. [52] showed that linear coding is sufficient to achieve the optimum max-flow from the source to the receivers. Note that we talk of the same flow to all the receivers. While some receivers may have a high max-flow min-cut capacity, others may have lower capacities. All that is guaranteed for network coding is that we achieve the minimum of these possible flows, *but* to all receivers simultaneously, something that multicast routing cannot achieve. More recently this has come to be called *homogeneous multicast capacity* to distinguish it from the general case, where one may attempt to send different messages to the various receivers, such as in scalable video coding, where we would seek to perform *heterogeneous multicast*. We have seen that MD-FEC and its variations can be used to achieve heterogeneous multicast. In the next section, it will be combined with network coding for greater efficiency.

Practical Network Coding

Practical network coding (PNC) was introduced by Chou et al. [53] to allow network coding to be used in real networks where there are random delays, packet losses, and variable link bandwidths. It is based on the random network coding of Ho et al. [54], which showed that random linear codes can come very close to achieving homogeneous multicast capacity, albeit with a certain small probability of nondecodability due to the unlikely occurrence of linearly dependent codewords. However, it has been shown, if the code symbols are uniformly chosen independently from Galois field $GF(2^8)$, that the probability of this nondecodability is quite small (i.e., less than or equal to 2^{-8}). Now, the finite field $GF(2^8)$ corresponds to 8-bit symbols or bytes and is easily integrated into real networks. In PNC the code vectors must be transmitted along with the data as a packet header, resulting in a small inefficiency for typical packet sizes around 1500 Bytes. As a packet moves through the network from source to receiver, the coefficient header gets updated consecutively as it passes through each intermediate node where the random combination of incoming packets will be repeated. Another key aspect of PNC is the *generation* concept: only packets in the current generation can be linearly coded together. For our work in video application of this idea, we have chosen the generation size to be one GOP, typically one-half second. This does introduce a delay issue as the packets are held in an intermediate node's buffer for a certain time to accumulate before the output starts from that node.

Example 13.4–3: Practical Network Coding

Here, we present a toy example of PNC using $GF(2^2)$, consisting of just four elements, 0,1,2, and 3. We start with four data packets of (coincidentally) four elements each and code them with random coefficient vector [2 3 0 1], as shown in Figure 13.4–10.

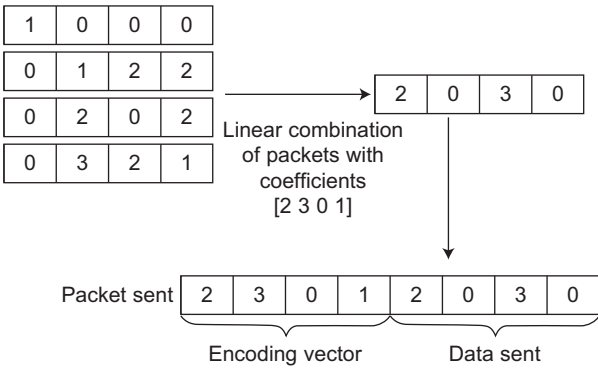
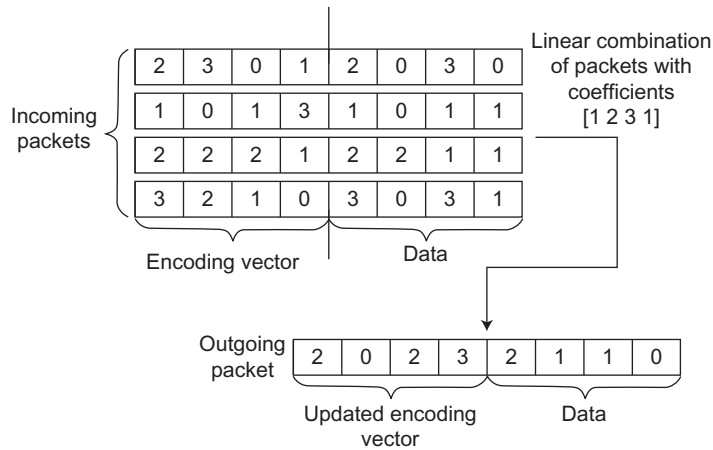


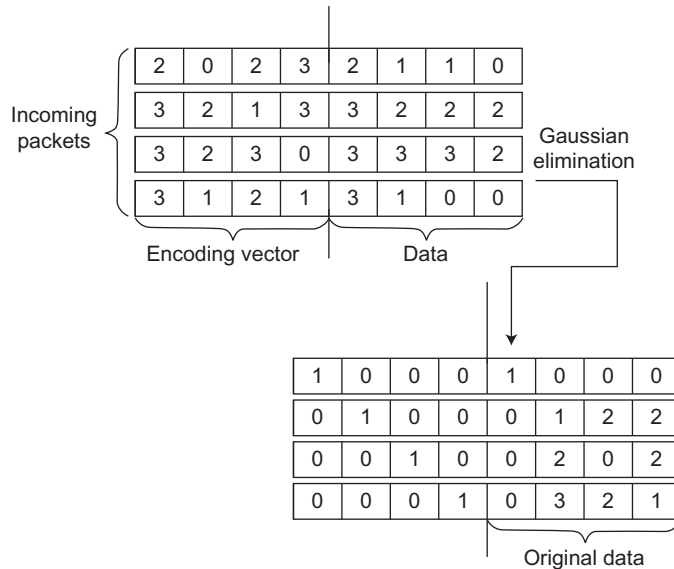
FIGURE 13.4–10

PNC computations at source.


FIGURE 13.4–11

PNC computation at intermediate node.

Figure 13.4–11 shows how four network-coded packets are linearly combined with random coefficient vector $[1 \ 2 \ 3 \ 1]$ at an intermediate node. Finally, Figure 13.4–12 shows four network-coded packets arriving at a receiver and their solution via Gaussian elimination [55] on the augmented matrix, outputting the original data plus an identity matrix header in place of the coefficient vectors.


FIGURE 13.4–12

PND decoding at receiver using Gaussian elimination.

The addition and multiplication tables for $GF(2^2)$ are provided in end-of-chapter problem 9, where you are asked to verify the calculations in [Figures 13.4–10](#) through [13.4–12](#). Note the overhead due to packet header is large here (50%) due to the short length of the data packets in this simple example. ■

PNC Together with MDC

Network coding can achieve a very efficient homogeneous multicast, but if our receivers have different needs and maxflow bandwidths, then we need to augment PNC with some means to get the right information to the right place. One such method is layered coding [\[56\]](#), wherein a scalable bitstream is broken into a base layer and 2–3 optional enhancement layers. The packets in these layers can then be addressed appropriately to get to the right users, and the optimal way of doing this would be to construct the required multicast trees relying on complete knowledge of the network. A much simpler method would be to incorporate MD-FEC methods to allow the quality of the received video to vary with the total number of packets received. Such a method was indicated by Chou et al. [\[53\]](#) and developed more fully in [\[57\]](#). The method herein called MD-PNC can be understood with reference to [Figure 13.4–13](#), where across the top we see a scalable bitstream broken into a number of progressive layers. Below this in the figure, we see that this bitstream has been mapped onto a set of packets (descriptions), but zero symbols have been inserted in place of the parity symbols of an FEC code. To understand how this can work, we must remember that the linear network coding is vertical across the symbols. So in

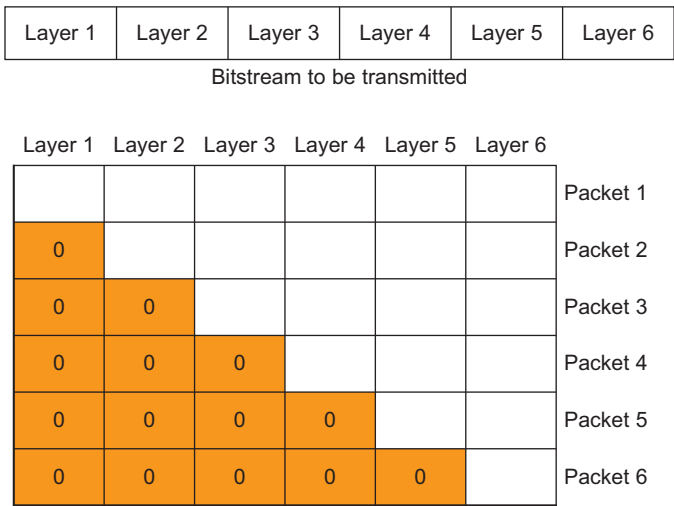


FIGURE 13.4–13

Simple illustration of the MD-PNC concept.

the first column, only layer 1 data will be present. Thus if any receiver obtains one or more of the network-coded versions of these packets, it will (only with high probability, because of the random code) be able to decode layer 1 in its Gaussian elimination procedure. Similarly, any two packets will only contain information on layers 1 and 2, and so these layers can be successfully decoded at a receiver that receives any two packets. Continuing in this way, we eventually see that MD-PNC behaves similarly to MD-FEC and so the distortion equation (13.2–1) will be the same, again with high probability. However, with MD-PNC some channel rate must be reserved for the packet headers containing the linear combination coefficients. Thus these coefficient headers must be added to the packets shown in Figure 13.4–13. The rate equation (13.2–2) is thus modified on the right-hand side to $B' < B$ because of this header. With MD-PNC, no addresses need be present on the packets, as all the packets are randomly linear combined at each network node and then sent out at the capacity of each outbound link.

Example 13.4–4: MD-PNC

A comparison of routing (unicast), routing with replication (multicast), and MD-PNC was presented in [57] for the simple butterfly net with link bandwidths of 500 Kbps and loss rates of 20%, 10%, and 1%. The Foreman clip was encoded by enhanced MC-EZBC at CIF resolution at 30 fps, and the PSNR was optimized in place of MSE distortion. The optimal multicast results were obtained with two paths for each receiver, one with a bandwidth of 500 Kbps and the second with a bandwidth of 250 Kbps. The results for 20% loss rate are shown in Figure 13.4–14. For this simple network, we see a substantial improvement due to MD-PNC versus both unicast and multicast. Video clip results are available at this book's Web site.

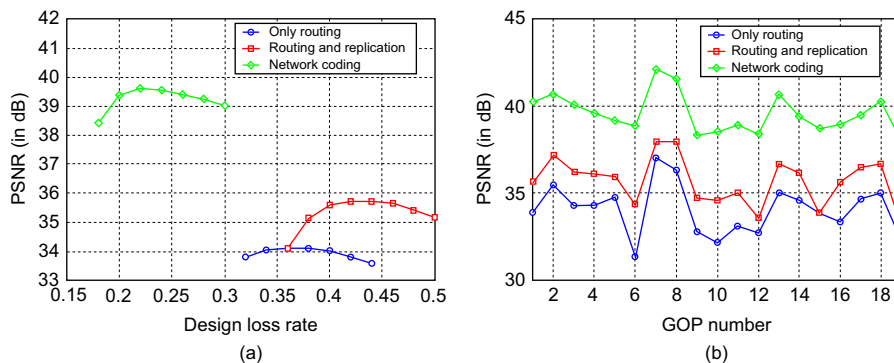


FIGURE 13.4–14

The foreman clip PSNRs with a link-loss rate of 20%: (a) variation of PSNR versus design loss rate, (b) PSNR versus frame number at best-design loss rate.

CONCLUSIONS

Compressed video data must be protected and adapted to the channel in the case of lossy transmission. This chapter has introduced the concept of error resilience of video coders and their use in transport-layer error protection for wired and wireless channels, a key difference being the significant probability of packet bit errors in the latter. We have shown examples of scalable video coders together with an multiple description coding (MDC) and introduced an extension of MD-FEC to data adaptation useful for heterogeneous networks. We presented network coding as an improvement over multicast routing that can achieve homogeneous network capacity. Finally, we presented a combination of multiple descriptions and practical network coding called MD-PNC, and showed its usefulness for video transmission on networks.

A nice overview of video communication networks is contained in Schonfeld [58]. An overview of JSCC for video communications is contained in Zhai et al. [59].

PROBLEMS

1. This problem concerns the sync word concept, the problem being how to segment a VLC bitstream into the proper characters (messages). One way to do this, together with Huffman coding, is to add an additional low probability (say $0 < p \ll 1$) message to the message source called *sync*. If we already had a set of independent messages m_1, m_2, \dots, m_M with probabilities p_1, p_2, \dots, p_M , comprising a message source with entropy H , show that for sufficiently small p , the total entropy including the sync word is less than $H + \epsilon$, for any given $\epsilon > 0$. Hint: Assume that the sync word constitutes an independent message, so that the *compound source* uses an original message with probability $(1 - p)p_i$ or a sync word with probability p . Note that

$$\lim_{p \searrow 0} p \log \frac{1}{p} = \lim_{n \nearrow \infty} 2^{-n} \log 2^n = \lim_{n \nearrow \infty} n 2^{-n} = 0.$$

2. In [Example 13.1–1](#), the outputs of the even and odd scalar quantizers are the two descriptions to be coded.
 - (a) For description 1, the output of the even quantizer, what is the entropy (rate) and distortion, assuming an input random variable uniformly distributed on $[0, 10]$.
 - (b) Compare the total rate for the two descriptions to the entropy rate for the original single-description quantizer, assuming the same random variable input.

3. Assume that bit errors on a certain link are independent and occur at the rate, i.e., with probability 1×10^{-3} . Assume that symbols (messages) are composed of 8-bit bytes. What then is the corresponding byte error rate? Assume that packets are 100 bytes long. What then is the corresponding packet error rate? Repeat for 300-byte packets.
4. Express the values α_k in terms of the source rate breakpoints R_k and the channel bandwidth B for MD-FEC as presented in (13.2-1) and (13.2-2).
5. This problem is about calculating the total number of bits needed for MD-FEC.
 - (a) Refer to Figure 13.2-9. Assume that R_1 through R_4 are expressed in bits. Calculate the total number of bits (both information bits and parity bits) needed for the four packets (descriptions), in terms of R_1 through R_4 .
 - (b) Now assume the goal is to create N packets. Calculate the total number of bits needed for N packets, in terms of R_1, R_2, \dots, R_N .
6. In channel coding theory, *code rate* is defined as the ratio of information bits to the total (information + parity) number of bits.
 - (a) Using the results from the previous problem, what is the code rate of MD-FEC for N packets, expressed in terms of R_1, R_2, \dots, R_N ?
 - (b) Based on the definition of code rate, any nontrivial code must have a rate in the interval $(0, 1]$. Let N be given. Assuming that $R_k, k = 1, 2, \dots, N$ can be real numbers, show that MD-FEC can achieve any rate $r \in [\frac{1}{N}, 1]$. In other words, given a number $r \in [\frac{1}{N}, 1]$, find R_1, R_2, \dots, R_N , such that the rate of the corresponding MD-FEC code is r .
7. In Example 13.4-1 on digital item adaptation (DIA), two coded videos were bitrate adapted (reduced) as shown in Figure 13.4-4. These figures show a slight decrease in PSNR performance at the second transcoding (adaptation), which uses only six quality layers. Would there be any further losses in quality at subsequent bitrate adaptations that may happen further on in the network?
8. One GOP consists of MC-EZBC-encoded bitstreams $\{Q_{lt}^{MV}, Q_{lt,ls}^{YUV}\}$, where $\{1 \leq l_t \leq 5\}$ and $\{1 \leq l_s \leq 6\}$ (refer to Section 13.4 for notation). For simplicity, we set the size $|Q_1^{MV}| = 4000$, $|Q_2^{MV}| = 1000$, $|Q_3^{MV}| = 700$, $|Q_4^{MV}| = 300$, and $|Q_5^{MV}| = 0$, all in bytes. The sizes of the sub-bitstream partitions $\{Q_{lt}^{MV}, Q_{lt,ls}^{YUV}\}$ are given in Table 13.4-1.
 - (a) What is the size of the adapted GOP if it is adapted to half resolution and one-quarter frame rate?
 - (b) Suppose the original bitstream is encoded with FGA-FEC as in Figure 13.4-5. Each motion vector sub-bitstream is encoded with a 1/4 code rate FEC code, and each YUV sub-bitstream is encoded with a 1/2 code rate FEC code. What is the size of the adapted GOP with FEC if the GOP is again adapted to half resolution and one-quarter frame rate?

Table 13.4–1 Sub-bitstream Sizes					
$l_t \setminus l_s$	1	2	3	4	5
1	100	100	200	100	200
2	200	200	400	400	500
3	800	800	1200	1000	1000
4	2200	2400	3000	2200	2000
5	4400	5300	5800	3200	2400
6	4600	6400	5000	2400	2000

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Galois field arithmetic table

FIGURE 13.P–1

Arithmetic tables for $GF(2^2)$.

9. The addition and multiplication tables for $GF(2^2)$ are shown in Figure 13.P–1. Use them to verify the computations in the toy example of Figures 13.4–10 through 13.4–12.

REFERENCES

[1] A. S. Tenenbaum, *Computer Networks*, 3rd. Ed., Prentice-Hall, Upper Saddle River, NJ, 1996.

[2] J. F. Kurose and K. W. Ross, *Computer Networking*, 3rd Ed., Pearson–Addison Wesley, Boston, MA, 2005.

[3] IETF Differentiated Services Working Group, homepage. Available at <http://www.ietf.org/html.charters/diffserv-charter.html>.

[4] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The Macroscopic Behavior of the TCP Congestive Avoidance Algorithm,” *Computer Comm. Review*, vol. 27, no. 3, July 1997.

[5] J. Padhye, J. Kurose, and D. Towsley, *A TCP-friendly Rate Adjustment Protocol for Continuous Media Flows Over Best Effort Networks*, U. Mass. CMPSCI Tech Report. 98–04, October 1998.

[6] V. Paxson, “End-to-End Internet Packet Dynamics,” *IEEE/ACM Trans. Networking*, vol. 7, no. 3, pp. 277–292, June 1999.

[7] M. Ghanbari, *Video Coding: An Introduction to Standard Codecs*, IEEE Telecomm. Series, London, UK, 1999.

- [8] Y. Takashima, M. Wada, and H. Murakami, "Reversible Variable Length Codes," *IEEE Trans. Comm.*, vol. 43, no. 243, pp. 158–162, February 1995.
- [9] J. Wen and J. Villasenor, "A Class of Reversible Variable Length Codes for Robust Image and Video Coding," *Proc. IEEE ICIP 1997*, pp. 65–68, Santa Barbara, CA, October 1997.
- [10] N. Farber, T. Wiegand, and B. Girod, *Error Correcting RVLC*, ITU/VCEG Q15-E-32, British Columbia, CA, July 1998.
- [11] B. Girod, "Bidirectionally Decodable Streams of Prefix Code-Words," *IEEE Comm. Letters*, vol. 3, no. 8, pp. 245–247, August 1999.
- [12] V. A. Vaishampayam, "Design of Multiple Description Scalar Quantizers," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 821–834, May 1993.
- [13] Y. Wang, M. Orchard, V. Vaishampayam, and A. R. Reibman, "Multiple Description Coding Using Pairwise Correlating Transform," *IEEE Trans. Image Process.*, vol. 10, no. 3, pp. 351–366, March 2001.
- [14] R. Puri and K. Ramchandran, "Multiple Description Source Coding Using Forward Error Correction Codes," *Proc. 33rd ACSSC*, Pacific Grove, CA, October 1999.
- [15] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in Wireless Environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 657–673, July 2003.
- [16] J. B. Anderson, *Digital Transmission Engineering*, IEEE Press, Piscataway, NJ, 1999.
- [17] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley and Sons, New York, 1968.
- [18] J. G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd Ed., Prentice-Hall, Upper Saddle River, NJ, 2002.
- [19] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, December 1997.
- [20] T. Rappaport, A. Annamalai, R. M. Buehrer, and W. H. Tranter, "Wireless Communications: Past Events and a Future Perspective," *IEEE Comm. Magazine*, 50th Anniv. Commem. Issue, no. 5, pp. 148–161, May 2002.
- [21] J. W. Modestino and D. G. Daut, "Combined Source-Channel Coding of Images," *IEEE Trans. Comm.*, vol. COM-27, no. 11, pp. 1644–1659, November 1979.
- [22] M. Bystrom and J. W. Modestino, "Combined Source-Channel Coding Schemes for Video Transmission over an Additive White Gaussian Noise Channel," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 6, pp. 880–890, June 2000.
- [23] W. Zeng and B. Liu, "Geometric Structure Based Error Concealment with Novel Applications in Block-Based Low-Bit-Rate Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 4, pp. 648–665, June 1999.
- [24] W. M. Lam, A. R. Reibman, and B. Liu, "Recovery of Lost or Erroneously Received Motion Vectors," *Proc. ICASSP*, vol. 5, pp. 417–420, March 1993.
- [25] Y.-K. Wang, M. M. Hannuksela, V. Varsa, A. Hourunranta, and M. Gabbouj, "The Error Concealment Feature of the H.26L Test Model," *Proc. ICIP*, vol. 2, pp. 729–732, Rochester, NY, September 2002.
- [26] A. K. Katsaggelos and N. K. Galatsanos, Eds., *Signal Recovery Techniques for Image and Video Compression*, Kluwer Academic Publishers, 1998.
- [27] N. S. Jayant, "Subsampling of a DPCM Speech Channel to Provide Two 'Self-Contained' Half-Rate Channels," *Bell Syst. Tech. J.*, vol. 60, no. 4, pp. 501–509, April 1981.
- [28] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1983.

- [29] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1988.
- [30] I. V. Bajić and J. W. Woods, "Maximum Minimal Distance Partitioning of the \mathbb{Z}^2 Lattice," *IEEE Trans. Inform. Theory*, vol. 49, no. 4, pp. 981–992, April 2003.
- [31] Y. Wang and D.-M. Chung, "Robust Image Coding and Transport in Wireless Networks Using Non-Hierarchical Decomposition," *Mobile Multimedia Communications*, (D. J. Goodman and D. Raychaudhury, Eds.), Plenum Press, 1997.
- [32] C. D. Creusere, "A New Method of Robust Image Compression Based on the Embedded Zerotree Wavelet Algorithm," *IEEE Trans. Image Processing*, vol. 6, no. 10, pp. 1436–1442, October 1997.
- [33] I. V. Bajić and J. W. Woods, "Domain-Based Multiple Description Coding of Images and Video," *IEEE Trans. Image Processing*, vol. 12, no. 10, pp. 1211–1225, October 2003.
- [34] J. K. Rogers and P. C. Cosman, "Wavelet Zerotree Image Compression with Packetization," *IEEE Signal Processing Letters*, vol. 5, no. 5, pp. 105–107, May 1998.
- [35] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.
- [36] I. V. Bajić, "Adaptive MAP Error Concealment for Dispersively Packetized Wavelet-Coded Images," *IEEE Trans. Image Processing*, in press.
- [37] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority Encoding Transmission," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1737–1744, November 1996.
- [38] R. Puri, K.-W. Lee, K. Ramchandran, and V. Bharghavan, "An Integrated Source Transcoding and Congestion Control Paradigm for Video Streaming in the Internet," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 18–32, March 2001.
- [39] A. E. Mohr, E. A. Riskin, and R. E. Laedner, "Unequal Loss Protection: Graceful Degradation Over Packet Erasure Channels Through Forward Error Correction," *IEEE J. Select. Areas Commun.*, vol. 18, no. 6, pp. 819–828, June 2000.
- [40] B. G. Kim and W. A. Pearlman, "An Embedded Wavelet Video Coder Using Three-Dimensional Set Partitioning in Hierarchical Trees," *Proc. Data Compress. Conf. (DCC)*, pp. 251–260, Snowbird, Utah, March 1997.
- [41] I. V. Bajić and J. W. Woods, "EZBC Video Streaming with Channel Coding and Error Concealment," *Proc. SPIE VCIP*, vol. 5150, pp. 512–522, Lugano, Switzerland, 2003.
- [42] S. Kumar, L. Xu, M. K. Mandal, and S. Panchanathan, "Overview of Error Resiliency Schemes in H.264/AVC Standard," *J. on Visual Communications and Image Representation*, vol. 17, no. 2, pp. 425–450, April 2006.
- [43] M. Karczewicz and R. Kurceren, "The SP and SI Frames Design for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 637–644, July 2003.
- [44] *MPEG-21 Overview V.4*, ISO/IEC JTC1/SC29/WG11, MPEG2002/N4801, 2002.
- [45] *MPEG-21 Digital Item Adaptation WD (v3.0)*, ISO/IEC JTC1/SC29/WG11, MPEG2002/N5178, 2002.
- [46] S.-F. Chang and A. Vetro, "Video Adaptation: Concepts, Technologies, and Open Issues," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 148–158, January 2005.
- [47] S.-T. Hsiang and J. W. Woods, "Embedded Video Coding Using Invertible Motion Compensated 3-D Subband Filter Bank," *Signal Processing: Image Comm.*, vol. 16, no. 8, pp. 705–724, May, 2001.

- [48] Hewlet-Packard Labs. Search term, “Structured Scalable Meta-formats (SSM)”. Available at <http://www.hpl.hp.com/techreports>.
- [49] Y. Shan, I. V. Bajić, S. Kalyanaraman, and J. W. Woods, “Joint Source-Network Error Control for Scalable Overlay Video Streaming,” *Proc. IEEE ICIP 2005*, Genoa, IT, September 2005.
- [50] Y. Shan, I. V. Bajić, J. W. Woods, and S. Kalyanaraman, “Scalable Video Streaming with Fine-Grain Adaptive Forward Error Correction,” *IEEE Trans. Cir. and Sys. for Video Technology*, vol. 19, no. 9, September 2009.
- [51] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network Information Flow,” *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1215, July 2000.
- [52] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear Network Coding,” *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, February 2003.
- [53] P. Chou, Y. Wu, and K. Jain, “Practical Network Coding,” *Proc. 41st Allerton Conf. on Comm., Control, and Comp.*, Monticello, IL, October 2003.
- [54] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, “The Benefits of Coding over Routing in a Randomized Setting,” *Proc. IEEE Intl. Sympos. Inform. Theory*, Yokohama, Japan, July 2003.
- [55] http://en.wikipedia.org/wiki/Gaussian_elimination.
- [56] S. McCanne and M. Vetterli, “Joint Source/Channel Coding for Multicast Packet Video,” *Proc. of IEEE ICIP*, vol. 1, 1995.
- [57] A. K. Ramasubramonian and J. W. Woods, “Video Multicast Using Network Coding,” *Proc. SPIE Visual Comm. and Image Process. (VCIP)*, San Jose, CA, January 2009.
- [58] D. Schonfeld, “Video Communications Networks,” Chapter 9.3 in *Handbook of Image and Video Processing*, 2nd Ed., A. C. Bovik Editor, Elsevier/Academic Press, Burlington, MA, 2005.
- [59] F. Zhai, Y. Eisenberg, and A. G. Katsaggelos, “Joint Source-Channel Coding for Video Communications,” Chapter 9.4 in *Handbook of Image and Video Processing*, 2nd Ed., A. C. Bovik Editor, Elsevier/Academic Press, Burlington, MA, 2005.

Index

Page numbers followed by n indicates a footnote.

- 1-D continuous time windows, 154–156
- 1-D Fourier transform, 21
 - rectangular pulse function, 16
- 1-D Kalman filter, 270
 - equations, 272–273
 - LMMSE estimator, 271
 - observation equation, 270
 - signal state equation, 271
 - system matrix, 271
 - See also* 2-D Kalman filter
- 2-D ARMA model, 406
- 2-D discrete-space Fourier transform. *See* 2-D Fourier transform
- 2-D discrete-space systems, 9
 - linear system, 10
 - shift invariance, 10
 - simple systems, 9–10
- 2-D filter stability, 92
 - first quadrant support, 94, 96
 - fourth quadrant support, 96, 97
 - LSI system, 92–93
 - NSHP, 100–102
 - ROC, 95–96, 97
 - root mapping, 98–100
 - second quadrant support, 95, 96
 - third quadrant support, 96
 - See also* 2-D systems
- 2-D Fourier continuous transform, 27
 - coordinate system rotation, 28
 - inverse, 27
 - projection-slice theorem, 29–30
 - rotational invariance, 28–29
 - See also* 2-D Fourier transform (2-D FT)
- 2-D Fourier transform (2-D FT), 13
 - complex plane wave, 20
 - convergence types, 15
 - Fourier convolution theorem, 19–20
 - hexagonal sampling lattice, 52
 - inverse, 16, 18–19
 - line impulse, 16
 - operator properties, 20–21
 - pairs, 21–22
 - properties, 14
 - rectangular pulse function, 15–16, 17
 - separable operator, 14
 - separable signal, 24–25
 - symmetry properties, 25–26
 - Z-transform comparison, 91–92
 - See also* 1-D Fourier transform; Continuous-space Fourier transform
- 2-D FT. *See* 2-D Fourier transform
- 2-D innovations sequence, 268
- 2-D Kalman filter, 274
 - error-covariance equations, 275
 - gain array, 275
 - global state vector, 274
 - scalar filtering equations, 275
 - See also* 1-D Kalman filter
- 2-D random field. *See* Two-dimensional random field
- 2-D raster manner, 507
- 2-D rectangular sinc function
- 2-D recursive filter design, 168
 - error criteria, 169–170
 - Prony example, 173
 - space-domain design, 170–173
 - See also* Fully recursive filter design (FRF design)
- 2-D signal. *See* Two-dimensional signal
- 2-D systems
 - input/output equation, 75
 - recursion direction, 76
 - simple difference equation, 76–79
 - stability in, 12–13
 - See also* 2-D filter stability
- 2-D Z-transform, 79–80
 - calculation, 82–83
 - closed-form solution, 89–90
 - convolution, 86
 - Fourier transform comparison, 91–92
 - general case, 84–85
 - intersection, 81
 - inverse, 88
 - linear mapping of variables, 87–88
 - properties, 85–86
 - zero loci, 80–81
 - See also* Region of convergence (ROC); 2-D Fourier transform
- 2-D vector, 149–150
- 2K containers in Digital Cinema, 464
- 3-D difference equation, 396
- 3-D filters, 396
- 3-D Fourier transform, 395, 403
 - interframe filtering, 405
 - inverse, 395

3-D Fourier transform (*Continued*)
 properties, 395–396
 3-D Kalman-reduced support regions, 410
 3-D linear shift invariant system, 393
 3-D linear system, 393, 396
 convolution, 394, 396
 separable operator, 396
 3-D perspective plot, 17
 zoomed-in contour plot, 17
 3-D rhombic dodecahedron, 55
 3-D sampling, 398–399
 3-D sampling theorem, 397–398
 3-D signal, 393, 415
 3D-SPIHT, 548–549
 3-D transform coder, 467
 4-D matrix, 150, 309n
 in image restoration, 308
 Kalman gain operator, 456
 4K containers for digital cinema, 464
 4:2:0 specification, 463
 1080i, 463

A

a posteriori estimation, 289
 CGM, 292
 clique system, 291–292
 Gauss-Markov image model, 289
 line field modeling edge, 292
 line field potential values, 293
 noncausal Markov random field, 290
 SA, 293–298
 A/D converter. *See* Uniform quantization
 AC. *See* Arithmetic coding
 Acknowledgement retransmission scheme (ACK/NACK scheme), 376
 Adaptive directional lifting transform (ADL transform), 373
 coded with 5/3 SWT, 375
 quadtree and directions, 374
 Additive white Gaussian noise channel (AWGN channel), 539
 Additive-increase multiplicative-decrease, 532
 ADL transform. *See* Adaptive directional lifting transform
 Advanced Television System Committee (ATSC), 463, 485
 broadcast formats, 463
 Advanced Video Coder (AVC), 504
 AEP. *See* Asymptotic equipartition principle
 Affine motion model, 430
 Alias error, 48
 2000 × 1000-pixel image, 49
 in directional case, 50
 zoomed-in section, 49

Anti-alias filtering, 184
 Aperture, 421
 effect, 214, 422
 problem, 421–422
 AR random signal model. *See* Autoregressive random signal model
 Arithmetic coding (AC), 347–349
 EZBC context-based, 368–369
 ARMA. *See* Autoregressive moving average
 ARQ. *See* Automatic repeat request
 Asymptotic equipartition principle (AEP), 386–387
 See also Entropy
 ATSC. *See* Advanced Television System Committee
 Automatic repeat request (ARQ), 536
 Autoregressive moving average (ARMA), 402
 Autoregressive random signal model, 263
 optimal linear prediction, 264–265
 QP predictor, 265
 Available bandwidth, 523
 AVC. *See* Advanced Video Coder
 Average
 MSE distortion, 355
 quantizer distortion, 337
 value entropy, 346
 AWGN channel. *See* Additive white Gaussian noise channel

B

Backward motion compensation, 479
 Bartlett window, 155
 Bayer array, Bayer filter, 213
 Bayesian, 289, 447
 detection/estimation, 454
 MC prediction, 448–450
 model, 520
 motion estimation and segmentation, 450–453
 motion/segmentation, 519, 521
 segment and displacement potential, 451
 Bessel function
 first-order, 23
 zeroth-order, 23
 Best-efforts network, 530–531
 BIBO. *See* Bounded-input bounded-output
 Binary digits (bits), 329
 Bit errors, 532
 Bitrate, average, 352
 bits. *See* Binary digits (bits)
 Block matching (BM), 423
 hierarchical, 426
 MSE in, 424
 problems with, 425–426
 PSNR performance, 425
 three-step, 424

Blurred SNR (BSNR), 279
 BM. *See* Block matching
 Boundary-matching algorithm, 539
 Bounded-input bounded-output (BIBO), 12, 92
 for 2-D systems, 12
 LSI system stability, 92–93
 NSHP Filter Stability, 100–101
 Box filter, 223
 FIR, 223
 Fourier transform, 224
 magnitude frequency response, 224
 See also Image processing
 Brightness. *See* Luminance
 BSNR. *See* Blurred SNR
 Butterfly network, 562, 563

C

Carrier-to-noise ratio (CNR), 493
 Cathode ray tube (CRT), 214, 217
 camera output voltage, 218
 gamma value, 217
 interlaced, 400
 Causal Markov concepts, 291
 CBR. *See* Constant bitrate
 CCD. *See* Charge-coupled device
 CCFL. *See* Cold cathode fluorescent lamp
 CDF. *See* Cumulative distribution function
 CDF filter. *See* Cohen-Daubechies-Feauveau filter
 CFA. *See* Color filter array
 CGM. *See* Compound Gauss-Markov
 Channel capacity, 383
 Channel coder, 329, 547
 design with video coder, 537
 digital image communication system, 330
 Channel coding rate, 538
 Charge-coupled device (CCD), 66, 195, 212
 sensor noise, 214
 Chromaticity, 205
 D65 standard illuminant, 207
 of standard observer, 206
 CIE. *See* Commission Internationale de L'eclairage
 CIELAB, 210
 CIF. *See* Common intermediate format
 Circular windows, 154, 156
 Cleanup pass, 370
 Clique system, 291, 447
 first-order, 291–292
 CMOS. *See* Complementary metal oxide semiconductor;
 Coupled metal-oxide semiconductor
 CNR. *See* Carrier-to-noise ratio
 Cohen-Daubechies-Feauveau filter (CDF filter), 181
 comparison with QMF, 182
 frequency response, 182
 step response, 183
 Cold cathode fluorescent lamp (CCFL), 216
 Color
 background adaptation, 205, 206
 chromaticity values, 205, 206–207
 color-matching functions, 204, 205
 difference components, 463
 display color primaries, 208
 human eye, 203
 linear model equations, 203, 204
 object wavelength spectrum, 207
 sensitivity functions, 203
 sensor response functions, 207, 208
 spectral radiance functions, 209
 white balance, 208
 Color filter array (CFA), 213, 456
 Color image coding, 370
 scalable coder results, 372
 subsampling strategies, 371
 See also JPEG 2000 standard (JP2K standard)
 Color space, 209
 CIELAB, 210
 ITU Recommendation 709, 210–211
 sRGB, 211–212
 Commission Internationale de L'eclairage (CIE), 194
 Common intermediate format (CIF), 462
 Complementary metal oxide semiconductor (CMOS), 66
 Compound Gauss-Markov (CGM), 292
 Compressed video sensitivity, 522–523
 Computed tomography (CT), 29
 Conditional replenishment, 477
 Conjugate antisymmetric part, 26
 Conjugate symmetric part, 25, 26
 Constant bitrate (CBR), 467
 Constant-coefficient. *See* Linear constant-parameter
 Constraint equation for motion, 421
 Continuous wavelet transform (CWT), 141
 Continuous-space Fourier transform, 27
 nonisotropic signal spectra, 44–47
 spatial frequency aliasing, 39
 Continuous-space IFT
 hexagonal sampling lattice, 52
 reconstruction formula, 41
 Contrast, 196
 adaptation, 199
 Contrast sensitivity function (CSF), 197, 198
 3-D, 202
 linear amplitude plot, 199
 spatiotemporal, 201
 standard observer, 198
 temporal, 201–202

Contribution quality, 477
 Control points for motion, 430
 Convergence
 generalized function, 15
 mean-square, 15
 uniform, 15
 Convex hull, 358
 Convolution
 2-D, 10, 11–12, 86
 3-D, 394
 methods, sectioned, 145–146
 property, 21, 86
 representation, 394, 396
 Convolution theorem
 2-D Z-transform, 86
 Fourier, 19–20
 Correlation function, 262
 2-D random field, 258
 geometric, 324
 random sequence, 322
 two-parameter, 259
 WSS (wide sense stationary), 323
 Cosine wave, 8–9
 Coupled metal-oxide semiconductor (CMOS), 195
 Bayer filter, 214
 multilayer, 213
 sensor noise, 214
 Covariance function
 2-D random field, 258
 random field calculation, 260
 random sequence, 322
 wide-sense homogeneity, 259
 CRT. *See* Cathode ray tube
 CSF. *See* Contrast sensitivity function
 CT. *See* Computed tomography
 Cumulative distribution function (CDF), 319
 See also Distribution function
 CWT. *See* Continuous wavelet transform

D

D5. *See* International Telecommunication Union (ITU): 601
 digital SDTV
 D65 standard illuminant, 207
 Data array, 331
 Data partition (DP), 533, 551
 Data service node (DSN), 556
 dB. *See* Decibels
 DC value, 21
 DCDM. *See* Digital cinema distribution master
 DCI formats, 463–464
 DCT. *See* Discrete cosine transform
 Dead zone, 469
 DeCarlo–Strintzis theorem, 99
 Decibels (dB), 425
 Decimation in frequency (DIF), 144
 Decimation in time (DIT), 144
 Decimation matrix, 67
 Decoder
 3-D multidimensionally stable, 401
 digital image communication system, 330
 failure, 375
 in image coder, 365
 M-JPEG 2000 standard, 476
 Degraded video restoration, 453
 Bayesian approach, 454
 in blotch-type artifacts presence, 454
 observation model, 453
 Deinterlacer, 441
 conventional, 442–443
 median, 443–445
 motion-compensated, 445–446, 447
 Deinterlacing, 440, 441
 See also Deinterlacer
 Delay property
 DFS, 112, 114–115
 DFT, 119
 FT operator, 21
 Z-transform, 86
 Demosaicing, 456–459
 Density domain. *See* Contrast
 DFD. *See* Displaced frame difference
 DFS. *See* Discrete Fourier series
 DFT. *See* Discrete Fourier transform
 DIA engine. *See* Digital item adaptation engine
 Diamond subsampling
 effect on frequency, 68–69
 sublattice, 67–68
 DIF. *See* Decimation in frequency
 Differential entropy, 387
 Differential pulse-code modulation (DPCM), 330n, 334
 2-D DPCM coder, 334
 motion-compensated, 479
 property, 335
 spatiotemporal generalization, 478
 Differentiated services (DiffServ), 531
 Digital
 image communication system, 330
 items, 555
 still camera, 212
 Digital cinema distribution master (DCDM), 464
 Digital image compression, 329
 2-D DCT, 331–333
 2-D DPCM coder, 334
 dyadic subband decomposition, 334

- source coder, 329
- SWT, 333–334
- Digital item adaptation engine (DIA engine), 555, 556
- Digital light processing (DLP), 216
- Digital video compression, 467
 - compressed video sensitivity, 522–523
 - interframe coding, 477, 480, 486–487, 503
 - intraframe coding, 468
 - nonlocal intraprediction, 517
 - object-based coding, 519–522
 - scalable video coders, 493
- Digital video formats, 461
 - ATSC broadcast formats, 463
 - CIF, 462
 - DCI formats, 463–464
 - ITU 601 digital SDTV, 462–463
 - SIF, 462
- Digital video processing, 415
 - Bayesian method, 447
 - degraded video restoration, 453
 - digital video formats, 461
 - interframe processing, 415–421
 - motion estimation and compensation, 421
 - motion-compensated filtering, 434
 - super-resolution, 455
- Directional transforms, 372
- Discrete cosine transform (DCT), 109, 125
 - 1-D case, 126–127
 - 2-D, 331–333
 - 2-D symmetric extension, 131–132
 - basis function image, 126
 - directional DCT, 373
 - fast DCT methods, 144–145
 - inverse, 126, 127, 128
 - MATLAB function, 128, 129
 - properties, 129–131
 - and SWT, 141
 - See also* Discrete Fourier series (DFS); Discrete Fourier transform (DFT); Subband/wavelet transform (SWT)
- Discrete cosine transform (DCT) coders, 350
 - AC coefficients, 351
 - bitrate, average, 352
 - Block-DCT Coding, 352–354
 - image, 353, 354
 - quantization matrix, 351
 - total MSE, 352
 - UTQ, 351
- Discrete Fourier series (DFS), 109–110
 - amplitude part, 112
 - calculation, 111
 - delay property, 114–115
 - inverse, 110–111
 - periodic convolution, 113–114
 - properties, 111, 112–113
 - See also* Discrete Fourier transform (DFT); Discrete cosine transform (DCT); Subband/wavelet transform (SWT)
- Discrete Fourier transform (DFT), 109, 115
 - 2-D circular convolution, 120–121
 - basis functions, 116
 - circular convolution, 120
 - comparison with DCT, 129
 - fast DFT algorithm, 143–144
 - Fourier transform relationship, 122
 - interpolation, 124–125
 - ones on diagonal of square, 117–118
 - properties, 118–120
 - sampling effect in frequency, 123–124
 - symmetry in real-valued signal, 122–123
 - See also* Discrete Fourier series (DFS); Discrete cosine transform (DCT); Subband/wavelet transform (SWT)
- Discrete wavelet transform (DWT), 142
 - See also* Subband/wavelet transform (SWT)
- Dispersive packetization (DP), 540
 - demonstration, 547
 - extension to three dimensions, 544
 - images, 540–544
 - of motion-compensated SWT video, 545
 - performance, 546
 - source sample splitting, 541
 - SWT, 542, 543
 - video, 544–545
- Displaced frame difference (DFD), 437, 492, 539
- Distortion rate, 359
- Distribution function, 319
 - properties, 320
 - test set, 343
 - See also* Random variables; Cumulative distribution function (CDF)
- Distribution print, 215
- DIT. *See* Decimation in time
- D–log E curve, 215, 216
- DLP. *See* Digital light processing
- Dominant pass, 364
- Double tree algorithm, 362
- Doubly stochastic Gaussian (DSG), 297
- Downsampling, 58
 - 2 × 2 decimation, 60–61
 - anti-alias filters, 184
 - general, 67
 - rectangular, 58, 59
 - system element, 59
- DP. *See* Data partition (DP); Dispersive packetization
- DPCM. *See* Differential pulse-code modulation
- DSG. *See* Doubly stochastic Gaussian

DSN. *See* Data service node

DV codec, 472–474

DV coder, 467, 472

DV macroblock, 473

Feed-forward or look-ahead strategy, 473

DWT. *See* Discrete wavelet transform

Dyadic decomposition, 333, 354

binary codeword calculation, 381

subband/wavelet structure, 362

E

EBCOT. *See* Embedded block coding with optimal truncation (EBCOT)

ECSQ. *See* Entropy-coded scalar quantization

ECVQ. *See* Entropy-coded vector quantization

Edge detection, 235

on smoothed noisy image, 239

Sobel filter outputs, 236, 238

thresholded output, 237

See also Object detection

Edge linking, 238

gray-level image, 240

noisy gradient image, 240

SEL, 238, 240

Z-J search algorithm, 239n

Eigenvector, DCT, 130–131

Electronic image sensors

Bayer array, 213

CCD, 212, 213, 214

CFA, 213, 214

CMOS, 212, 213, 214

CRT, 214

sensor fill factor, 214–215

See also Film

Electronic projectors, 216

Elementary streams (ES), 485

EM. *See* Expectation-maximization

Embedded block coding with optimal truncation (EBCOT), 369

ADL transform, 373

means, 370

Embedded zero block coder (EZBC), 362, 367

algorithm, 367–368

context modeling for AC, 369

context-based AC, 368–369

Embedded zero-tree wavelet (EZW), 362, 363

algorithm, 364–365

extractor, 365

precoder, 365

trees of coefficients, 363

Embedding, quantizer bins for, 361

End of block (EOB), 375, 470

End of slice (EOS), 482

Energy function, 291, 447, 451

joint pdf, 292

motion field prior model, 451–452

neighboring clique's potential function, 447

Entropy, 384, 386

bounds on, 382

chain rule for, 382–383

coder, 330, 331

differential, 387–388

See also Kraft inequality; Information theory; Mutual information; Random variables

Entropy coding, 346

Arithmetic coding (AC), 347–349

ECSQ, 349–350

ECVQ, 249

error sensitivity, 350

Huffman coding, 346–347

VLC, 350

Entropy-coded scalar quantization (ECSQ), 349

joint quantizer and entropy encoder, 350

Entropy-coded vector quantization (ECVQ), 349

EOB. *See* End of block

EOS. *See* End of slice

Equal slope condition, 358

Eric image

contour plot, 4, 5

intensity plot, 4, 6

mesh plot, 4, 5

Error

control coding, 534

covariance equations, 272, 273, 275, 411

resilient coding, 533–534

sensitivity, 350

Error concealment, 522, 538

in decoder, 537

dispersive packetization, 540

in H. 264/AVC, 554

MC, 549

nonnormative, 539

ES. *See* Elementary streams

Euler's equality, 15

Even quantizer, 534

Event, 319

Expectation-maximization (EM), 300

FoGSM model, 304, 305–306

GSM, 302

image identification and restoration, 298–302

pairwise densities, 305

Exp-Golomb codes, 533

Extension source, 346

Extractor, 365

EZBC. *See* Embedded zero block coder

EZW. *See* Embedded zero-tree wavelet

F

FEC. *See* Forward error correction
 FGA-FEC. *See* Fine-grain-adaptive-FEC
 Field of Gaussian scale mixtures model (FoGSM model), 304
 estimate, 305–306
 FIFO. *See* First-in first-out
 Film
 composition, 216
 digitization, 215
 distribution print, 215
 D–log E curve, 215, 216
 See also Electronic image sensors
 Filter design method, biorthogonal, 180
 power complementary pair, 181
 system frequency response, 181
 transfer function, 180
 Filter stability test, 99–100, 102
 Fine quantization, 338
 Fine-grain-adaptive-FEC (FGA-FEC), 556
 2-D SWT with, 140–141
 3-D adaptation, 561
 coding scheme, 560
 JSNC vs. random packet drop, 561
 overlay network, 557
 PSNR performance comparison vs. bitrate, 559
 video scalability dimensions, 557
 Finite impulse response (FIR), 92, 223, 401
 box filter, 223
 Finite impulse response filter design (FIR filter design), 153
 1-D continuous time windows, 154–156
 by 1-D filter transformation, 160–163
 using MATLAB, 156–159
 POCS, 166–168
 transform lowpass filter example, 164–166
 window function design, 153–154
 See also Infinite impulse response filter design (IIR filter design)
 Finite-state scalar quantization (FSSQ), 492
 FIR. *See* Finite impulse response
 FIR filter design. *See* Finite impulse response filter design
 First-in first-out (FIFO), 532
 First-order temporally causal model, 408
 Flexible macroblock ordering (FMO), 551
 mapping of macroblocks, 552
 Flicker method, 194–195
 FMO. *See* Flexible macroblock ordering
 FoGSM model. *See* Field of Gaussian scale mixtures model
 Forward error control coding, 535–536
 Forward error correction (FEC), 534
 Forward motion compensation, 490
 bidirectional MCTF variation, 500
 PSNR results, 494
 Fourier convolution theorem, 19–20

Fovea, 203
 fps. *See* Frames per second (fps)
 Fractional passes, 369, 370
 Frame
 based model, 402
 memories, 401
 rate conversion, 440
 repeat, 440, 441
 Frames per second (fps), 468
 FRF design. *See* Fully recursive filter design
 FSSQ. *See* Finite-state scalar quantization
 Fully embedded, 361
 SWT coders, 362–363
 Fully recursive filter design (FRF design), 174
 row sequences, 174
 row-operator form, 174
 SHP Wiener filter design, 176–177
 stability, 176
 system function, 175
 Future-present-past diagram, 408

G

Gain array, 411
 Gaussian estimation, inhomogeneous
 image model, 282
 RUKF estimate, 283, 284–285
 Wiener filter construction, 283–284
 Gaussian filter, 224–225
 Gaussian rate-distortion function, 389–390
 Gaussian scale mixture (GSM), 303
 Gibbs distributions, 291
 Gibbs model, 289, 448
 GOB. *See* Group of blocks
 Golomb-Rice codes, 533
 GOP. *See* Groups of picture
 Group of blocks (GOB), 486
 Groups of picture (GOP), 480
 GSM. *See* Gaussian scale mixture

H

H. 262. *See* MPEG: MPEG 2
 H. 263 coder, 486
 H. 264/AVC, 504, 550, 554
 allowed MV block sizes, 505
 data partitioning, 551
 directional prediction modes, 506
 flexible macroblock ordering, 551–552
 intra block refreshing, 554
 PSNR vs. bitrate, 506
 reference frame selection, 553
 switching frames, 552–553
 syntax, 551
 system diagram, 504

- H. 264/MVC, 503, 515
 - inter-view prediction hierarchy, 516
 - multiview test clip Ballroom, 517
 - H. 264/SVC, 503, 508
 - hierarchical B frame, 510, 511
 - for spatiotemporal scalability, 512
 - Haar filter pair, 139–140
 - Hanning window, 155
 - HBM. *See* Hierarchical block matching
 - HD. *See* High definition
 - HDR. *See* High dynamic range
 - HDTV. *See* High-definition television
 - Hexagonal sampling lattice, 51
 - discrete-space FT, 52
 - hexagonal alias repeat grid, 54
 - periodicity matrix, 52, 53
 - reconstruction formula, 54–55
 - Hierarchical block matching (HBM), 426–427
 - motion estimation, 436, 437
 - multiresolution coder, 488
 - See also* Block matching (BM)
 - Hierarchical VSBM (HVSBM), 426
 - backward motion vectors, 499
 - refining and splitting process, 427
 - High definition (HD), 56
 - formats, 462
 - resolutions, 215
 - High dynamic range (HDR), 218
 - High reliability channel (HRC), 514
 - High-definition television (HDTV), 209, 484
 - MPEG 2, 485
 - nonlinear function, 211
 - High-resolution (HR), 311, 455
 - Hi-Vision, 484
 - Horizontal wave, 7
 - HR. *See* High-resolution
 - HRC. *See* High reliability
 - Huang stability theorem, 98, 99
 - Huffman coding, 346–347
 - Human eye, 203
 - color receptor's sensitivity functions, 203
 - dynamic range, 215
 - relative luminous efficiency, 193
 - sensitivity, 215
 - visual system response, 207
 - Human visual system (HVS), 193, 200
 - 3-D CSFs, 202
 - local adaptation property, 200
 - spatiotemporal CSF, 201
 - temporal CSF, 201–202
 - See also* Human eye; Image visual properties, still
 - HVSBM. *See* Hierarchical VSBM
 - Hybrid
 - coder, 478
 - decoder, 479
 - filter, 401
- I**
- ICC. *See* International Color Consortium
 - ICM. *See* Iterated conditional mode
 - IDCT. *See* Inverse discrete cosine transform
 - Ideal decimation, 61
 - 2×2 decimation, 62
 - frequency domain support, 63–64
 - LL subband, 62
 - Ideal filter, 20
 - Ideal interpolation, 65–66
 - Ideal lowpass filter
 - circular passband, 22–23
 - impulse responses, 24
 - rectangular passband, 21–22
 - IDFS. *See* Inverse discrete Fourier series
 - IDFT. *See* Inverse discrete Fourier transform
 - IDWT. *See* Inverse discrete wavelet transform
 - IETF. *See* Internet Engineering Task Force
 - I-frame, 403n, 481
 - IFT. *See* Inverse Fourier transform
 - IIR. *See* Infinite impulse response
 - Image
 - coding robustness, 375–376
 - enhancement, 228–234
 - estimation, 257, 278–279
 - exercises, 316–318, 376–379
 - restoration, 257, 279–281
 - superresolution, 311–314
 - Image analysis, 223, 235
 - edge detection, 235–238
 - edge linking, 238–240
 - segmentation, 239, 340
 - See also* Image: enhancement
 - Image identification and restoration, 298
 - EM algorithm approach, 298–302
 - subband EM restoration, 303
 - See also* Expectation-maximization (EM); Non-Bayesian image estimate
 - Image processing
 - box filter, 223–224
 - color, 314–315
 - digital, 257
 - downward directed vertical axis, 76n
 - Gaussian filter, 224–225
 - Laplacian filter, 227–228
 - MATLAB toolbox function, 158, 165

- monochrome, 257
- Prewitt operator, 225
- raster scan, 169
- Sobel operator, 226–227
- See also* Image analysis
- Image visual properties, still, 195
 - contrast adaptation, 199, 200
 - contrast sensitivity function, 197, 198–199
 - Weber's law, 196–197
 - See also* Human visual system
- Imaging distortion, 44
- IMC. *See* Inverse motion compensation
- Improved-Definition Television, 417–418
- Impulse train, modulated, 42–43
- Indicator function, 22
- Infinite impulse response (IIR), 401
 - filter design, 153, 168–173
 - FRF design, 174–177
 - See also* Finite impulse response (FIR)
- Infinite observation domain, 266
 - SNR, 266
 - Wiener filter, 266, 267
- Information, 346
 - measure, 379, 380
 - mutual, 383–384
- Information theory, 341
 - bounds on entropy, 382
 - chain rule for entropy, 382–383
 - continuous sources, 387
 - data compression, 384
 - differential entropy, 387
 - discrete message source, 381–382
 - Gaussian random variable, 388
 - information measure, 379
 - Kraft inequality, 384–385
 - mutual information, 383–384
 - Shannon, Claude, 380
 - source coding theorem, 386–387
 - source entropy, 380–381
 - See also* Entropy; Kraft inequality; Rate-distortion theory; Source: coding theorem
- Initial value, 21
- Interframe
 - ARMA signal model, 405
 - video coders, 467
 - Wiener filter, 405, 406
- Interframe coding, 477
 - 1-D DPCM to interframe coding, 477–478
 - backward motion compensation, 479
 - coding standards, 480, 503
 - H. 263 coder, 486
 - H. 264/AVC, 504–507
 - H. 264/MVC, 515–517
 - H. 264/SVC, 508–512
 - hybrid decoder, 479
 - MC spatiotemporal prediction, 478–480
 - motion-compensated DPCM, 479
 - MPEG 1, 480–482
 - MPEG 2, 482–484
 - MPEG 3, 484–485
 - MPEG 4, 485
 - MPEG-coded bitstreams, 485–486
 - video coder mode control, 507–508
- Interframe filtering, 404
 - using 3-D Fourier transforms, 405
 - ARMA signal model, 405
- Interframe processing, 415
 - spatiotemporal RUKF, 415–417
 - visually optimized filter, 419–421
- Interframe SWT coders, 486–487
 - 3-D subband filter, 487
 - Frame-rate conversion, 440
 - invertibility conditions, 490
 - Karlson's filter tree frequency decomposition, 488
 - MC-deinterlacing, 445
 - MC-Kalman filter, 436
 - MCTF, 490, 491
 - MC-Wiener filter, 435
 - motion-compensated SWT hybrid coding, 487–489
 - Motion compensated processing, 434
 - multiresolution coder, 488
 - spatiotemporal transform coding, 489–492
- Interlaced CRT displays, 400
- Interlaced display, source, 217
- Interlaced video, 57, 58, 400
 - 3-D sampling, 400–401
 - CRT, 217
 - diamond sampling, 37
 - MPEG 2, 483
 - sampling in 2-D vertical-temporal domain, 57
- International Color Consortium (ICC), 210
- International Commission on Illumination. *See* Commission Internationale de L'éclairage (CIE)
- International Standards Organization (ISO), 480
- International Telecommunication Union (ITU), 210, 482
 - inverse transformation, 210
 - linear transformation, 210
 - nonlinear color space, 211, 601
 - digital SDTV, 462–463
- Internet Engineering Task Force (IETF), 531
- Internet protocol (IP), 529
 - networks, 530–532
- Interpolation function, 42, 43, 125
- Intra macroblock displacement compensation, 517

Intrablock refreshing, 554
 H. 264/AVC, 554
 Intraframe ARMA signal model, 403
 Intraframe coding, 468
 4:1:1 color, 473
 DV codec, 472–474
 DV macroblock, 473
 intraframe SWT coding, 474–476
 M-JPEG, 467, 470–471
 M-JPEG 2000, 476–477
 M-JPEG pseudo algorithm, 469–470
 with rate control, 469
 SWT coding, 474–476
 SWT filter study, 475–476
 Inverse 2-D Fourier transform, 16
 proof, 18
 Inverse 3-D Fourier transform, 395
 Inverse discrete cosine transform (IDCT), 127, 128
 Inverse discrete Fourier series (IDFS), 110–111
 Inverse discrete Fourier transform (IDFT), 117
 Inverse discrete wavelet transform (IDWT), 142
 Inverse Fourier transform (IFT), 18
 3-D sampling theorem, 397
 continuous-space, 52
 discrete-space FT, 52
 rectangular sampling, 38
 symmetric frequency response, 22
 Inverse motion compensation (IMC), 435
 Inverse subband/wavelet transform (ISWT), 138, 141
 Inverse Z-transform, 88, 89
 2-D polynomials don't factor, 90–91
 long division method, 90
 See also Inverse Fourier transform (IFT)
 IP. *See* Internet protocol
 Iris, 203
 ISNR. *See* SNR improvement
 ISO. *See* International Standards Organization
 Isolated zero (IZ), 364
 ISWT. *See* Inverse subband/wavelet transform
 Iterated conditional mode (ICM), 447
 ITU. *See* International Telecommunication Union
 IZ. *See* Isolated zero

J

JND. *See* Just-noticeable difference
 Johnston's filter 16C, 179–180
 Joint motion estimation and segmentation, 450–453
 Joint source channel coding (JSCC), 537–538
 Joint source–network coding (JSNC), 555
 fine grain adaptive FEC, 556–562
 MPEG 21, digital item adaptation, 555–556
 video network coding, 562–563

Joint video team (JVT), 504
 JPEG 2000 standard (JP2K standard), 369, 372
 PSNR for, 372
 scanning, 370
 SWT coded image, 374
 JSCC. *See* Joint source channel coding
 JSNC. *See* Joint source–network coding
 Just-noticeable difference (JND), 195
 on local background brightness, 200
 local contrast adaptation, 199
 threshold, 197
 JVT. *See* Joint video team

K

Kaiser window, 155
 11×11 impulse response, 157–158, 159, 160
 circular, 158
 contour plot, 157
 magnitude response, 156, 159
 See also Finite impulse response filter design (FIR filter design)
 Kalman filter, 274
 estimate, 273
 Filtering estimate, 272
 See also Image processing
 Kalman gain vector, 272, 273
 Kalman smoother estimates, 273
 Karhunen-Loeve transform (KLT), 131, 330
 K-means algorithm, 242
 256×256 cameraman image, 243
 class indices, 244
 class means, 245, 246
 for color images, 245, 246
 convergence, 243, 244
 optimal values, 242–243
 Kraft inequality, 384–385.
 See also Entropy; Information theory

L

L^2 norm, 169
 LAGRANGE multiplier, 349
 Laplace rate-distortion function, 390
 Laplacian filter, 227–228
 See also Image processing
 Lapped orthogonal transform (LOT), 331
 block-DCT transformation, 332
 SWT, 333
 Lattice, 66
 diamond-shaped sampling, 56
 hexagonal sampling, 51
 Layers and protocol stack, 531
 LBG algorithm. *See* Linde, Buzo, and Gray algorithm

- LCCDE. *See* Linear constant coefficient difference equation
- LCD display. *See* Liquid crystal digital display
- LCOS. *See* Liquid crystal on silicon
- Least-squares solution, 343
- LED light sources, 216–217
- LeGall/Tabatabai (LGT), 487
- Lens, 203
- LGT. *See* LeGall/Tabatabai
- Linde, Buzo, and Gray algorithm (LBG algorithm), 341, 343
- Line field modeling edge, 292
- Line field potential values, 293
- Line impulse
- in discrete space, 2
 - Fourier transform, 16
- Line system, 486n
- Linear
- constant-parameter, 393–394
 - mapping, 87–88
 - prediction, optimal, 264–265
- Linear constant coefficient difference equation (LCCDE), 75
- Linear filtering, 228
- in intensity domain, 230–231, 232
 - MSE, 228, 230
 - noise sailboat image, 228, 229
 - output after 3×3 box filter, 229–230
 - See also* Median filtering; Image analysis
- Linear minimum mean-square error (LMMSE), 271
- Linear shift-invariant system (LSI system), 10
- estimation, 277–279
 - filter, 277
 - frequency response, 20
 - impulse response, 11
 - inhomogeneous Gaussian estimates, 285
 - restoration, 279–282
 - stability, 12–13, 92–93
- Linear space-variant (LSV), 302
- Linear spatial systems. *See* 2-D systems
- Linear time-invariant systems (LTI systems), 13
- Linearity property
- DCT, 129
 - DFS, 112
 - DFT, 118
 - FT operator, 21
 - Z-transform, 85
- LIP. *See* List of insignificant pixels
- Liquid crystal digital display (LCD display), 216, 217
- spectral radiance functions, 209
- Liquid crystal on silicon (LCOS), 216
- LIS. *See* List of insignificant sets
- List of insignificant pixels (LIP), 365
- List of insignificant sets (LIS), 365
- List of significant pixels (LSP), 365
- LMMSE. *See* Linear minimum mean-square error
- Long division method, 90
- Lossless coder, 329
- Lossy coder, 329
- LOT. *See* Lapped orthogonal transform
- Low reliability (LRC), 514
- Lowpass filter (LPF), 44
- Low-resolution (LR), 311
- LPF. *See* Lowpass filter
- LR. *See* Low-resolution
- LRC. *See* Low reliability
- LSI system. *See* Linear shift-invariant system
- LSP. *See* List of significant pixels
- LSV. *See* Linear space-variant
- LTI systems. *See* Linear time-invariant systems
- Luminance, 193
- CIE, 194
 - flicker method, 194–195
 - light, 195
 - luminous efficiency, 193
 - for sensors, 195
 - split-screen approach, 194
- ## M
- MAD. *See* Mean absolute difference
- MAE. *See* Mean absolute error
- Magnitude
- only approximation, 169
 - and phase design, 170
 - refinement, 370
- Magnitude transfer function (MTF), 195
- Main profile, at High level (MP@HL), 485
- M-algorithm, 297
- Markov random field sequence, 406, 407
- Markov random sequence covariance matrix, 131
- Markov-generating kernel values, 304
- Maximum a posteriori (MAP) estimate, 293
- Maximum likelihood estimate, 3112n
- Maximum transmission unit (MTU), 532
- Maximum-likelihood (ML), 299, 312, 539
- MC. *See* Motion compensation
- MCTF. *See* Motion-compensated temporal filter
- MDC. *See* Multiple description coding
- MD-FEC. *See* Multiple description forward error correction
- MD-PNC, 566.
- See also* Practical network coding (PNC)
- MDSP. *See* Multidimensional signal processing
- Mean absolute difference (MAD), 423n
- Mean absolute error (MAE), 231, 423
- Mean field annealing (MFA), 447

- Mean square error (MSE), 228, 423
 - in block matching, 424
 - distortion, 338
 - of uniform quantization, 339
 - Median filtering, 231
 - example, 233
 - salt and pepper noise, 233, 234
 - See also* Linear filtering; Image analysis
 - Mesh-based methods, 430–434
 - control points, 430
 - example, 432–433
 - pseudo code, 431–432
 - regular triangular mesh grid, 431
 - message source, 381–382
 - MFA. *See* Mean field annealing
 - Midrise quantizer, 337
 - Midtread quantizer, 337
 - Minimum mean-square error (MMSE), 271, 297, 407
 - M-JPEG, 467, 470–471
 - M-JPEG 2000, 476–477
 - pseudo algorithm, 469–470
 - rate control, 471, 472
 - ML. *See* Maximum-likelihood
 - MM MC-RUKF. *See* Multimodel MC-RUKF
 - MMSE. *See* Minimum mean-square error
 - Model noise, 271, 282, 289, 299
 - Monotonic nondecreasing distribution function, 320
 - Mother wavelet, 141
 - Motion compensation (MC), 421, 506
 - forward, 494
 - generalization, 517
 - via global pan vector, 495
 - MC-EZBC, 502–503, 549
 - MC-Kalman filter, 436–437, 437–440
 - MC-Wiener filter, 435–436
 - in MPEG 1, 481
 - spatiotemporal prediction, 478–480
 - warping, 436
 - Motion estimation and compensation, 421
 - aperture problem, 422
 - background covering and uncovering, 422
 - block-matching, 423–426, 426–427
 - mesh-based methods, 430–434
 - optical flow methods, 429–430
 - overlapped block motion compensation, 427–428
 - pel-recursive motion estimation, 429
 - Motion model failure, 459
 - Motion-compensated filtering, 434
 - deinterlacing, 440
 - frame-rate conversion, 440
 - LSI filter modification, 434
 - MC-Kalman filter, 436–437
 - MC-Wiener filter, 435–436
 - along motion path, 435
 - Motion-compensated temporal filter (MCTF), 490, 496
 - adaptive LGT/Haar, 502
 - bidirectional, 498–502
 - four-level Haar filter, 490
 - Haar, 492, 496
 - LGT 5/3 filter, 491
 - scalable video coder, 495, 496–497
 - MP@HL. *See* Main profile, at High level
 - MPEG
 - coded bitstream video processing, 485–486
 - half-band filters, 462
 - MPEG 1, 480, 481, 482
 - MPEG 2, 482–484
 - MPEG 21, digital item adaptation, 555–556
 - MPEG 4, 485
 - See also* H. 264/AVC
 - MSB. *See* Most significant bit
 - MSE. *See* Mean square error
 - MTF. *See* Magnitude transfer function
 - MTU. *See* Maximum transmission unit
 - Multicast routing, 562
 - Multidimensional signal processing (MDSP), 6
 - Multimodel MC-RUKF (MM MC-RUKF), 437
 - experimental result, 438–440
 - noisy motion vectors problem, 437–438
 - Multiple description coding (MDC), 533
 - PNC with, 566–567
 - redundancy, 535
 - scalar quantizer, 533–534
 - Multiple description forward error correction (MD-FEC), 545–548, 549–550
 - Multiplication property
 - DFS, 112
 - DFT, 118
 - Fourier Transform (FT), 20
 - Multiresolution SWT coding, 359–361
 - Multiview coding (MVC), 523
 - Mutual information, 383
 - See also* Entropy
 - MVC. *See* Multiview coding
- ## N
- NAL. *See* Network abstraction layer
 - National Television System Committee (NTSC), 417
 - Neighborhood sets of the field log, 304
 - Network abstraction layer (NAL), 508, 551
 - Network adaptation, 508
 - Network coder, 529
 - Network coding, 562

Noise-thresholding operations, 286–289

Non-Bayesian image estimate, 306

- BM3D estimate, 308
- least squares, 308–310
- nonlocal means, 306–308
- total variation, 310–211

Noncausal Markov

- field regions for, 290–291, 407
- in first-order clique system, 291
- random field, 290

Nondyadic SWT decompositions, 362

Noninterlaced video, 400

- See* Progressive video

Nonlinear filtering. *See* Median filtering

Nonlocal intraprediction, 517

- intra macroblock displacement compensation, 517
- template matching, 518

Nonlocal means estimation, 306

Nonorthogonal sampling

- diamond-shaped sampling lattice, 56–57
- general hexagonal case, 53–54
- hexagonal sampling lattice, 51–53
- sampling efficiency, 55–56
- sampling matrix, 50
- See also* Rectangular sampling

Nonsymmetric half-plane (NSHP), 76, 100, 175, 297

- causal Wiener filter, 279–280
- coefficient array support, 101
- filter stability, 100, 101
- stability test, 102

Nonsymmetric half-space (NSHS), 408, 477

- random field sequence scanning, 409

NSHP. *See* Nonsymmetric half-plane

NSHS. *See* Nonsymmetric half-space

NTSC. *See* National Television System Committee

Null event, 319

O

O&A method. *See* Overlap-and-add

Object detection, 248

- object segmentation, 248–250
- template matching, 250–253
- See also* Edge detection

Object segmentation, 248

- in Miss America test clip, 249
- See also* Region segmentation

Object-based coding, 519

- hybrid MCP object video coder, 520
- object-based SWT coder, 521–522

OBMC. *See* Overlapped block motion compensation

Observation equation

- 2-D Kalman filter, 274
- 3-D Kalman, 409
- Gaussian estimation, 282

Occlusion problem, 435

OCSWT. *See* Overcomplete subband/wavelet transform

Octave decomposition. *See* Dyadic decomposition

Odd/even quantizer, 534

One-quarter spatial resolution, 557n

Open system interconnection (OSI), 531

Optical flow, 421, 426n

- methods, 429–430

OSI. *See* Open system interconnection

Overcomplete subband/wavelet transform (OCSWT), 286

Overlap-and-add (O&A method), 146

Overlapped block motion compensation (OBMC), 427–428

Overlay network, 555, 557

- adaptation, 560, 562
- DSN, 556

Oversampling camera, 66

P

Packet-based wired networks, 522, 544–545

Packetized zerotree wavelet method (PZW method), 542

Packet-loss rate, 532, 556

Padé approximation, 170–171

Parseval's theorem, 21

- DFS, 113
- DFT, 119

Partial differentiation in frequency, 21

pdf. *See* Probability density function

Peak SNR (PSNR), 286n, 425

- for JPEG 2000 test set images, 372
- for H.264/AVC, 506
- vs. packet loss, 543

Pel-recursive motion estimation, 429

Periodic convolution

- DFS, 112, 113–114
- DFT, 118

Periodic signal, 6–7, 15

- cosine wave, 8–9
- general periodicity, 7–8
- horizontal wave, 7

Periodicity, 7–8

- rectangular, 7

Periodicity matrix, 8, 52, 398

- in frequency domain, 52–53
- diamond sampling, 56
- rectangular sampling case, 53

Perspective plot. *See* Mesh plot

PET. *See* Priority encoding transmission

Plane wave propagation, 47–48
 pmf. *See* Probability mass function
 PNC. *See* Practical network coding
 POCS. *See* Projection onto convex sets
 Power spectral density (PSD), 262, 323, 403
 of image random field, 291
 model noise random field, 291
 spectral factors and, 267
 of typical image, 262
 Power spectrum, 262–263
 Practical network coding (PNC), 564–566
 computation at intermediate node, 565
 computations at source, 564
 with MDC, 566–567
 PND decoding, 565
 Precoder, 365
 Predictor estimate, one-step, 272
 Prefix code, 384
 binary, 385
 Prefix condition, 533
 Prewitt operator, 225
 See also Sobel operator
 Priority encoding transmission (PET), 546
 Probability density function (pdf), 259, 320, 341, 469
 Probability mass function (pmf), 293, 349
 Probability measure, 319
 Probability space, 319
 See also Random process; Random sequence
 Progressive video, 400
 See also Interlaced video
 Projection onto convex sets (POCS), 166
 algorithm, 167
 convergence, 167, 168
 Projection-slice theorem, 29–30
 Prony's method, 172–173
 Protocol stack, 531
 PSD. *See* Power spectral density
 PSNR. *See* Peak SNR
 PZW method. *See* Packetized zerotree wavelet method

Q

QCIF. *See* Quarter CIF
 QMF. *See* Quadrature magnitude filter (QMF); Quadrature mirror filter
 QoS. *See* Quality of service
 QP predictor, 265
 Quadrature magnitude filter (QMF), 139
 Quadrature mirror condition, 138
 Quadrature mirror filter (QMF), 177, 178
 Quality of service (QoS), 531
 Quantization, 335, 473
 in 2-D DPCM, 335

 data decorrelation, 334
 fine, 338
 fixed-length, 338
 matrix, 470, 351
 optimal MSE, 337, 338
 in Panter and Dite, 377
 scalar, 331, 487, 492
 in source-coding system, 330
 TCQ, 344
 uniform, 331, 337, 338, 339
 vector, 334, 345
 See also Entropy-coded scalar quantization (ECSQ);
 Entropy-coded vector quantization (ECVQ)
 Quantizer, 331, 335, 336
 average quantizer distortion, 337
 Central deadzone, 351n
 concatenation of, 360–361
 deadzone of, 337
 in differential pulse-code modulation, 330n
 embedded, 361
 index set, 346n
 model, 338, 351, 355
 MSE, 338–340
 rounds to the left, 336
 scalar MDC, 534
 scalar quantizer, 336
 scalar uniform, 350
 SQ, 336, 535
 SQ design algorithm, 340
 step size, 507
 uniform quantization, 338, 339
 UTQ, 351, 469
 vector quantization, 341
 See also Rate-distortion theory
 quantizer set, 359
 Quarter CIF (QCIF), 462, 484

R

Radon transform, 29
 Random field, 257
 Homogeneous, 258
 See also Random variable
 Random field sequence, 402
 3-D, 409
 homogeneous, 407
 Markov, 406
 Random process, 319, 324–325
 2-D random fields, 258
 distribution function, 319
 geometric correlation function, 324
 homogeneous, 403

- independent random variables, 321
- pdf, 320
- probability space, 319
- WSS, 323
- See also* Random sequence; Random variable
- Random sequence, 257, 319, 321–322
 - Gaussian noise, 259–260
 - homogeneous random field, 259
 - Markov, 131
 - second-order moments, 258–259
 - stationary, 322–323
 - wide-sense homogeneous (*See also* WSS), 259
 - WSS, 323
- Random variable, 319, 321
 - assumed, 337
 - discrete, 388
 - distribution function, 320
 - Gaussian, 388
 - independent, 321
 - Laplace, 377
 - Poisson, 214
 - scalar, 534
 - uniform, 338, 339
- Raster scan, 77, 100, 150n, 169, 248
- Rate-distortion theory, 331, 355, 388
 - Gaussian rate-distortion function, 389–390
 - Laplace rate-distortion function, 390
- Real valued signals, 26
 - DFT symmetry, 26, 122–123
- Real-time protocol (RTP), 530
 - at application layer, 532
- Receiver time-out time (RTO), 532
- Receiver-processing multidimensional filters, 418
- Reconstruction formula, 42
 - continuous-space IFT, 41
 - Haar SWTs, 140
 - for hexagonal sampling, 54–55
 - in ideal interpolation, 65
 - imaging distortion, 44
 - interpolation function, 43
 - modulated impulse train, 42–43
- Rectangular periodicity, 7
- Rectangular pulse function, 15–16
- Rectangular sampling, 37
 - alias error in images, 48–50
 - effect of, 45
 - Fourier sampling relation, 39
 - ideal, 44
 - nonisotropic signal spectra, 44–47
 - plane wave propagation, 47–48
 - reconstruction formula, 41–44
 - sample rate change, 58, 59
 - spatial frequency, 39–40
 - See also* Hexagonal sampling lattice
- Rectangular windows. *See* Separable windows
- Recursive subband decomposition. *See* Dyadic decomposition
- Reduced order model (ROM), 282
- Reduced update Kalman filter (RUKF), 276
 - approximate, 276–277
 - image estimation, 278–279
 - image restoration, 279–281
 - inhomogeneous Gaussian estimation, 283, 284–285
 - MC, 436, 437
 - MM MC-RUKF, 437–440
 - spatiotemporal, 415–417
 - steady-state, 277, 411
- Reed-Solomon codes (RS codes), 535
- Reference field, 483
- Reference frame selection (RFS), 553
- Region growing, 245, 246
 - 2-D complex magnitude plane, 82
 - 2-D filter stability, 95–96
 - algorithm, 246–247
 - gray-level 512×480 flower image, 247
 - from seed location, 247–248
- Region of convergence (ROC), 79–80, 81
 - unit step function, 83–84
 - See also* 2-D Z-transform
- Region segmentation, 239, 240
 - K-means algorithm, 242–245
 - manual histogram thresholding, 240–241
 - region growing, 245, 246–248
 - See also* Edge detection; Edge linking; Object segmentation
- Regularity problem, 143
- Replacement noise model, 453
- Resolution scalable coder (RSC), 494
 - average PSNR, 496
 - demonstration software, 495
- Retina, 203
- Reversible variable length coding (RVLCs), 533–534
- RFS. *See* Reference frame selection
- ROC. *See* Region of convergence
- ROM. *See* Reduced order model
- ROM Kalman filter (ROMKF), 282
- Root mapping, 81, 98
 - DeCarlo–Strintzis theorem, 99
 - filter stability test, 99–100
 - Huang theorem, 98, 99
- Rotated windows. *See* Circular windows
- Rotational invariance, 28, 30
- Round trip time (RTT), 532
- Row–column approach, 143
- Row-scanning order. *See* Raster scan

Row-transforms, 143
 RS codes. *See* Reed-Solomon codes
 RSC. *See* Resolution scalable coder
 RTO. *See* Receiver time-out time
 RTP. *See* Real-time protocol
 RTT. *See* Round trip time
 RUKF. *See* Reduced update Kalman filter
 RVLCS. *See* Reversible variable length coding

S

SA. *See* Simulated annealing
 SAD. *See* Sum absolute difference
 Sample rate change, 58, 66
 diamond subsampling, 67–69
 downsampling, 58–61
 general downsampling, 67
 ideal decimation, 61–64
 ideal interpolation, 65–66
 upsampling, 64–65
 Sampling efficiency, 55–56
 Sampling lattice, diamond-shaped, 56–57
 Sampling matrix, 50, 51, 53
 3-D, 398
 for 2-D interlaced data, 58
 diamond sampling, 56
 SANs. *See* Storage area networks
 Scalability, 493–496
 resolution, 367
 spatial, 511
 spatiotemporal, 512
 video scalability dimensions, 557
 See also Scalable Video Coder (SVC); Subband/wavelet (SWT) coders
 Scalable coder results comparison, 372
 Scalable SWT-based coders, 529
 Scalable Video Coder (SVC), 493, 508, 523
 bidirectional MCTF, 498–502
 coding comparison, 503
 covered pixels detection, 497–498
 enhanced MC-EZBC, 502–503
 frame output, 501
 MCTF, 496–497
 MPEG, 509
 quality scalability, 513
 SD and HD video, 494
 threaded hierarchical structure, 513
 unconnected blocks in flower garden, 499
 unidirectional motion field, 499
 video conferencing, 512–515
 Scalar Kalman filtering equations, 273
 Scalar quantization, 331, 341
 ECSQ, 349
 fine, 377
 FSSQ, 492
 interframe SWT coders optimization, 487
 Scalar quantizer (SQ), 336
 embedding, 361
 MDC, 534
 SD. *See* Standard definition
 SDI. *See* Spike-detector index
 SEL. *See* Sequential edge linking
 Sensor fill factor, 214–215
 Separability, 6, 112
 2-D DCT, 126, 132–133
 2-D FT, 18
 3-D, 394
 3-D LSI system, 394
 DFS, 112
 DFT, 118–119
 Separable operator, 14, 396
 DFS operator, 115
 Z-transform, 83
 Separable signal, 4–6, 394
 Fourier transform, 24–25
 FT operator, 21
 See also Periodic signal
 Separable windows, 154, 155, 156
 Sequential edge linking (SEL), 238
 Set partitioning in hierarchical trees coder (SPIHT coder), 365
 algorithm, 366
 parent–child dependencies, 365, 720p, 463
 Shank's method. *See* Prony's method
 Shanks theorem, 96
 Shannon, Claude, 380
 Shannon coding, 386
 Shape coding, 519
 Shift invariance, 10
 Shift property. *See* Delay property
 Shifting representation, 2, 10
 SHP. *See* Symmetric half-plane
 SHP Wiener filter design, 176
 magnitude, 176–177
 phase response, 177
 Side-match error, 539
 SIF. *See* Source interchange format
 Signal, 286
 2-D, 1, 30
 3-D, 393, 415
 ACK/NAK, 535
 AR, 270, 274, 298
 circularly bandlimited, 40
 decimated, 61, 67
 decomposition, 64
 discrete-space, 67

- error, 334, 335
- Gibbs-Markov, 447
- intraframe ARMA, 403
- isotropic, 135, 136
- model, 436
- nonisotropic, 44
- NTSC composite, 418
- periodic, 6–7, 8
- separable, 4, 21, 24
- shifting representation of, 2
- speech, 540
- state equation, 271
- step functions, 2
- symmetries, 25–26, 122
- upsampled, 64
- video, 415
- wedge support, 87n
- Signal spectra, nonisotropic, 44
 - continuous-space FT, 45
 - diagonal basic cell, 46
 - rectangular sampling effect, 45, 46
- Signal-to-noise ratio (SNR), 266
- Significant propagation, 370
- Simple difference equation
 - dimensioned vector equation, 79
 - LCCDE, 76
 - recursive calculation, 77–78
 - solution calculation, 77
- Simplified ++ stability test, 97–98, 99
- Simulated annealing (SA), 293
 - for image estimation, 294–295
 - for image restoration, 295–297
 - motion estimation, 447
 - parallel, 297–298
- Simulcast, 359–360, 517
 - solution, 510
- Singular value decomposition (SVD), 6
- Slice, 533, 554, 555
 - lengths, 522
- Slice-based packetization (SP), 545
 - and DP schemes on football sequence, 546
- SMPTE. *See* Society of Motion Picture and Television Engineers
- SMPTE D1. *See* International Telecommunication Union (ITU): 601 digital SDTV
- SNHC. *See* Synthetic natural hybrid coder
- SNR. *See* Signal-to-noise ratio
- SNR improvement (ISNR), 278
- Sobel filter
 - absolute value, 236
 - thresholded output, 236, 237
- Sobel operator, 226
 - contour plot, 227
 - edge detection, 235
 - imaginary part, 226
- Society of Motion Picture and Television Engineers (SMPTE), 463
- Solid-state display, 217, 400
- Source
 - coder, 329, 529, 547
 - coding theorem, 346, 386–387
 - decoder, 330
 - entropy, 380–381
- Source interchange format (SIF), 462
 - bitrates, 468
 - M-JPEG coding, 472
 - MPEG, 480, 482
- Source-coding system diagram, 330
- SP. *See* Slice-based packetization
- Space-domain design, 169, 170–171
 - Padé approximation, 170–171
 - Prony's method, 172–173
- Spatial convolution. *See* Convolution: 2-D
- Spatial domain aliasing, 125, 146
- Spatial frequency, 195n
 - complex exponential for, 21
 - contrast sensitivity, 197–199, 201
 - response
 - spatiotemporal CSF, 202
- Spatial Kalman filtering equations, 276
- Spatial signal. *See* Two-dimensional signal (2-D signal)
- Spatiotemporal frequency aliasing, 401
- Spatiotemporal constraint equation, 429
- Spatiotemporal filters, 401–406
- Spatiotemporal Markov models, 406
 - 3-D Kalman-reduced support regions, 410
 - causal and semicausal 3-D field sequences, 408–409
 - discrete Markov random field sequence, 406–408
 - first-order temporally causal model, 408
 - gain array, 411
 - local state vector, 410
 - reduced update spatiotemporal Kalman filter, 409
 - update region, 409, 410–411
- Spatiotemporal signal processing, 399
 - first-order temporal interframe Wiener filter, 406
 - interframe filtering, 404
 - interframe Wiener filter, 405
 - intraframe filtering, 402–403
 - intraframe Wiener filter, 403–404
 - spatiotemporal ARMA model, 402
 - spatiotemporal filters, 401
 - spatiotemporal sampling, 400–401
 - temporal average filter, 401

- Spatiotemporal transform coding, 489–492
- Spectral factorization, 267
 - 2-D innovations sequence, 268
 - MSE filter, 269
 - whitening filter realization, 268
- Spectral radiance functions, 209
- SPIHT coder. *See* Set partitioning in hierarchical trees coder
- Spike-detector index (SDI), 453
- Split-screen approach, 194
- SQ. *See* Scalar quantizer
- SR. *See* Superresolution
- sRGB color space, 209, 211–212
- Stack algorithm. *See* Zigangirov-Jelinek search algorithm (Z-J search algorithm)
- Standard definition (SD), 56, 215, 461, 468, 484
- Standard observer, 194, 198
 - chromaticity diagram, 206
 - CIE, 207
 - curve, 195
- Storage area networks (SANs), 529
- Strict-sense polynomial, 84
- Subband coding, 356–357
 - 3-D, 486
- Subband transform (*See* Subband/wavelet transform (SWT))
- Subband/wavelet domain estimation, 285
 - PSNR, 286n
 - signal, 286
 - SWT denoising, 286–289
- Subband/wavelet transform (SWT), 109, 133n, 333
 - 1-D, 140
 - ADL coded, 375
 - alias cancellation and reconstruction, 141
 - analysis/synthesis, 178, 182, 312n, 360
 - coder, 354
 - and DCT, 141
 - denoising, 286–289
 - directional filtering, 373
 - dyadic, 142
 - filter pairs, 177, 475–476
 - with FIR filters, 140–141
 - Fourier transform output, 135–136
 - frequency domain expressions, 134–135
 - fully embedded coders, 362
 - Haar filter pair, 139–140
 - hybrid coding, 487–489
 - interframe coding, 477, 486
 - intraframe coding, 474
 - inverse, 141, 286, 333
 - inverse 1-D, 138
 - lazy, 189
 - lifted, 189, 190, 373
 - motion-compensated hybrid coding, 487
 - multiresolution coding, 359
 - nondyadic decomposition, 362
 - object-based swt coder, 521–522
 - overcomplete, 286
 - rectangular, 133–134, 136
 - subband region reconstruction, 137
 - temporal output, 496
 - transform functions, 139
 - video coding, 540–550
 - and wavelet, 141–143
 - See also* Discrete Fourier series (DFS); Discrete Fourier transform (DFT); Discrete cosine transform (DCT)
- Subband/wavelet transform (SWT) robust
 - video coding, 540
- 3D-SPIHT, 548–549
- dispersive packetization, 540
- MC-EZBC, 549
- multiple description FEC, 545–548
- SWT values, 133
- Subband/wavelet transformation (SWT) coder, 354
 - constant slope conditions, 358
 - distortion rate, 355, 357
 - dyadic decomposition, 354
 - equal slope condition, 358
 - MC-SBC, 490
 - performance, 357
 - rate-distortion optimization, 355
 - subband coding, 356–357
- Subband/wavelet transformation (SWT) filter design, 177
 - anti-alias filtering, 184
 - biorthogonal filter design method, 180–181
 - CDF filter, 181–183
 - filter comparison, 358
 - Johnston's QMF design, 178–179
 - transfer function, 178
- Sublattice, 66
 - diamond, 67–68
- Subordinate pass, 364
- Sum absolute difference (SAD), 423n, 517
- Superresolution (SR), 312
 - image, 257, 311–314
 - type of, 312n
 - of video, 455–459
- Sure event, 319
- SVD. *See* Singular value decomposition
- Switching frames in H.264/AVC, 552–553
- SWT. *See* Subband/wavelet transform
- Symmetric convolution, 125
- Symmetric half-plane (SHP), 175
- Symmetry properties
 - DCT, 130

- DFS, 113
- DFT, 119–120
- Fourier transform, 25
- real-valued signals, 26
- Z-transform, 86
- Sync word, 522, 533n
- Syntax, 375, 551
- Synthetic natural hybrid coder (SNHC), 485
- T**
- TCP. *See* Transfer control protocol
- TCQ. *See* Trelliscoded quantization
- Template matching, 250–251, 518
 - BM, 423
 - for intra prediction, 518
 - SR, 312n
- Template matching spatial prediction (TMSP), 518
- Temporally causal, 408
 - Kalman filter, 455, 456
 - model, 408
 - totally ordered, 408, 409
- Temporally semicausal, 409
- Test model 5 (TM5), 484
- Texture, 519
 - coders, 485, 520, 521
 - HVS, 199
 - interlayer prediction model, 509
 - nonisotropic signal, 44
 - segmentation, 239
 - See also* Template matching
- Threading, 532
- Thresholding, manual histogram, 240
 - gray-level house image, 241
 - at minima, 241
- Time jitter, 532
- TM5. *See* Test model 5
- TMSP. *See* Template matching spatial prediction
- Tone mapping, 218
- Totally ordered temporally causal case. *See* Nonsymmetric half-space (NSHS)
- Trackable motion, 442
- Training set, 259, 343
- Transfer control protocol (TCP), 530, 536
 - friendly, 530, 532
 - at transport level, 532
- Transform block, 331, 332
- Transform filter design method
 - 1×1 order McClellan, 162
 - 9×9 lowpass filter, 162
 - magnitude response, 163
 - symmetric FIR filter, 160–161
 - See also* Transform lowpass filter
- Transform lowpass filter design, 164
 - example, 164
 - for image filtering, 164
 - McClellan transformation, 166
 - See also* 1-D filter transformation design
- Transformation, 330, 331
 - of 1-D filter, 160–166
 - in image coding, 330, 334
 - inverse, 210
 - invertibility conditions, 490
 - linear, 205, 210, 418
 - nonlinear, 210, 211, 231
 - orthogonal, 335
 - rate-distortion theory, 331
 - of variables, 87
 - See also* Information theory; Subband/wavelet transformation (SWT); Quantization
- Transformation and quantization, 335
- Transport stream (TS), 485, 508
- Transport-level error control, 535–536
- Trelliscoded quantization (TCQ), 344
- Triangular window. *See* Bartlett window
- TS. *See* Transport stream
- Two-dimensional random field (2-D random field), 257–258
 - AR signal models, 263–265
 - estimation, 265–266
 - filtering, 260–262
 - Gaussian noise, 259–260
 - homogeneous, 258–259
 - infinite observation domain, 266–267
 - NSHP causal Wiener filter, 279–280
 - power spectrum of images, 262–263
 - second-order moments, 258
 - spectral factorization, 267–269
 - wide-sense homogeneous, 259
- Two-dimensional recursive estimation
 - 1-D Kalman filter, 270–273
 - 2-D Kalman filtering, 274–275
 - approximate RUKF, 276–277
 - LSI estimation, 277–279
 - LSI restoration, 279–281
 - reduced order model, 282
 - RUKF, 276
 - steady-state RUKF, 277
 - See also* Gaussian estimation, inhomogeneous
- Two-dimensional signal (2-D signal), 1
 - 2-D convolution, 10–12
 - 2-D discrete-space systems, 9–10
 - contour plot of Eric, 4, 5
 - impulse, 1, 2
 - mesh plot, 4, 5

Two-dimensional signal (2-D signal) (*Continued*)

- periodic signals, 6–9
- separable signal, 4, 6
- stability, 12–13
- unit impulse line, 2, 3
- unit step function, 2, 3–4

U

- UDP. *See* User datagram protocol
- Unequal error protection (UEP), 551
- Uniform quantization, 337
 - MSE of, 339
 - subband coding, 356
 - of uniform random variable, 338, 339
- Uniform quantizer, 337
- Uniform threshold quantizer (UTQ), 351, 469
- Unit impulse line, 2, 3
- Unit step function
 - first-quadrant, 2, 31, 78, 83, 91, 104
 - fourth quadrant, 4, 83–84
 - ROC, 83–84
 - second, and third quadrants, 4
- Unitary matrix, 130
- Update region, 409, 410–411
 - covariance, 277, 411, 413
 - local, 276
- Upsampling, 64–65, 66
 - conventional deinterlacer, 441
 - lowpass filter for, 71
 - spatial, 257
- User datagram protocol (UDP), 532
- UTQ. *See* Uniform threshold quantizer

V

- Variable bitrate (VBR), 467, 468
- Variable length codes, 346, 533
- Variable length coding (VLC), 350
 - reversible, 533
- Variable-length code tree, 385
 - See also* Entropy; Information theory
- Variable-size block matching (VSBM), 426
- Variable-size mesh matching (VSMM), 433
- Variance function, 258, 282, 322
- VBR. *See* Variable bitrate
- VCEG. *See* Video Coding Experts Group
- VCL. *See* Video coding layer
- Vector quantization (VQ), 331, 341, 343–344
 - ECVQ, 349
 - on image, 344–345
 - least-squares solution, 343
 - optimality conditions, 342

PSNR vs. bits, 345

See also Linde, Buzo, and Gray algorithm (LBG algorithm)

- Video bitstream, 557, 558
 - DSNs, 558
 - MC-EZBC, 549
 - scalability, 493
- Video coder, 467
 - in application layer, 531
 - error-resilience, 529, 568
 - MC, 426
 - mode control, 507–508
- Video Coding Experts Group (VCEG), 503
- Video coding layer (VCL), 508, 551
- Video network coding, 562–563
 - PNC together with MDC, 566–567
 - practical, 564–566
- Video objects (VO), 485
- Video on demand (VOD), 486, 529
- Video on IP networks, 529
 - error concealment, 538
 - error-resilient coding, 533
 - multiple description coding, 534–535
 - overview, 530
 - transport-level error control, 535
 - wireless networks, 536
- Video scalability dimensions, 557
- Video super-resolution, 455
 - demosaicing, 456–459
- Video transmission, 529
 - exercises, 568–570
 - H. 264/AVC, 550
 - on IP networks, 529
 - joint source-network coding, 555
 - robust SWT video coding, 540
 - system diagram, 530
- Video typical bitrates, 468
- Visual scenes, 519
- VLC. *See* Variable length coding
- VO. *See* Video objects
- VOD. *See* Video on demand
- Voronoi regions, 343
- VQ. *See* Vector quantization
- VSBM. *See* Variable-size block matching
- VSMM. *See* Variable-size mesh matching

W

- Wavelength, 195n
- Wavelet coefficients, 143
 - See also* SWT values
- Wavelet decomposition. *See* Dyadic decomposition
- Wavelet packets, 362

Wavelet theory, 141, 142
 CWT, 141
 DWT, 142
 IDWT, 142
 regularity problem, 143
 Wavenumber, 47
 Weber's law, 196
 contrast, 196
 increments, 197
 intensity difference vs. background intensity, 196
 stimulus, 197
 White Gaussian noise, intensity-modulated, 214
 White noise, 260, 271
 2-D difference equation, 263
 exercises, 316, 378
 Gaussian, 305, 415
 low-resolution signal, 416
 random field sequence, 402, 409
 spatiotemporal, 403
 See also Signal-to-noise ratio (SNR)
 Whitening filter
 2-D, 267
 linear minimum MSE, 269
 realization of Wiener filter, 268
 Wide-sense stationary (WSS), 259, 323
 Wiener filter, 266, 267
 3-D, 435
 adaptive, 284
 blur restoration, 296
 CGM model, 295

 DFT-implemented, 281
 equation for 2-D noncausal, 266
 Gauss-Markov model, 295
 interframe, 405–406
 intraframe, 403–404
 MC, 435–436
 nonrecursive formulation, 271
 NSHP causal, 269–270
 SHP, 176
 simple, 283
 spatial, 301
 wavelet image denoising, 286–289
 whitening filter realization, 268
 Window function, 153
 classes, 154
 filter impulse response, 153
 properties, 154
 Wireless local area networks (WLANs), 529
 Wireless networks, 536
 joint source-channel coding, 537–538
 WLANs. *See* Wireless local area networks
 WSS. *See* Wide-sense stationary

Z

Zero loci, 80–81, 97
 Zero-phase design, 169
 Zero-tree root (ZTR), 363, 364
 Zigangirov-Jelinek search algorithm (Z-J search algorithm),
 239n
 ZTR. *See* Zero-tree root